OPIC
OFFICE DE LA PROPRIÉTÉ INTELLECTUELLE DU CANADA

CIPO
CANADIAN INTELLECTUAL PROPERTY OFFICE

(12) (19) (CA) **Demande-Application**

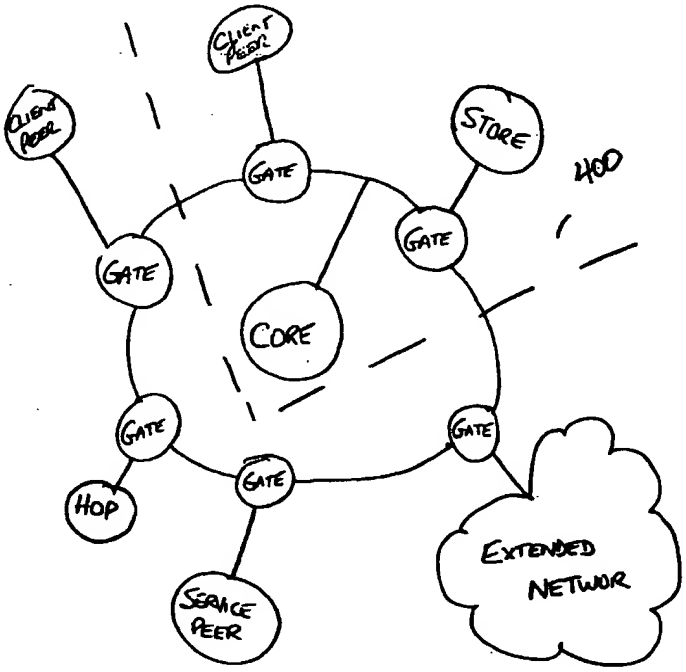
(21) (A1) **2,287,094**
(22) 1999/10/22
(43) 2000/04/22

(72) BUZZARD, GREGORY, US
(71) AT&T CORP., US

(51) Int.Cl.[6] H04L 12/16, H04L 12/14
(30) 1998/10/22 (60/105,195) US

(54) **METHODE ET APPAREIL D'ENREGISTREMENT A UN CERTAIN NOMBRE DE SERVICES INDEPENDANTS**

(54) **METHOD AND APPARATUS FOR PROVIDING A PROCESS FOR REGISTERING WITH A PLURALITY OF INDEPENDENT SERVICES**



(57) A method and an apparatus provide a user the capability of a single registration process or point of contact for registration which nonetheless enables the user to be authenticated to two or more independent service providers. A network architecture assumes the responsibility of functionality common to a plurality of service providers where such functionality includes service registration, authentication for a service, encryption for the communications with the service, monitoring of usage of the service and billing for such usage. This reduces the cost and complexity for the user and server end points of the network allowing a more diverse group of service providers to offer ever improving services without need for investing in processing structure that can be better incorporated within the network and shared amongst a plurality of providers.

ABSTRACT OF THE DISCLOSURE:

A method and an apparatus provide a user the capability of a single registration process or point of contact for registration which nonetheless enables the user to be authenticated to two or more independent service providers. A network architecture assumes the responsibility of functionality common to a plurality of service providers where such functionality includes service registration, authentication for a service, encryption for the communications with the service, monitoring of usage of the service and billing for such usage. This reduces the cost and complexity for the user and server end points of the network allowing a more diverse group of service providers to offer ever improving services without need for investing in processing structure that can be better incorporated within the network and shared amongst a plurality of providers.

METHOD AND APPARATUS FOR PROVIDING A PROCESS FOR REGISTERING WITH A PLURALITY OF INDEPENDENT SERVICES

Cross Reference to Related Applications

This application claims priority to provisional application 60/105,195 entitled "METHOD AND APPARATUS FOR PROVIDING A PROCESS FOR REGISTERING WITH A PLURALITY OF INDEPENDENT SERVICES" filed on October 22, 1998, the contents of which are incorporated herein by reference. The present application is related to provisional applications 60/105,189 entitled "METHOD AND APPARATUS FOR PROVIDING A PROCESS FOR REGISTERING WITH A PLURALITY OF INDEPENDENT SERVICES" filed on October 22, 1998, and provisional application 60/105,196 ENTITLED "A METHOD AND APPARATUS FOR ENHANCING NETWORK USAGE" filed on October 22, 1998.

Field of the Invention

The present invention is directed to a method and apparatus for providing network communications between users and service providers. More particularly, the present invention is directed to a method and apparatus by which a user can register with a network in a single registration process and become registered to a plurality of services provided by independent service providers.

Background of the Invention

In the past decade there has been an ever increasing use of communication networks to transfer information between users of network resources. Wireline

telephone networks have given way to wireless communications. Voice communications have given way to data communications. Data communications via groups of computer networks, such as the Internet, utilizing a commonly acceptable protocol permit communication capabilities unheard of in years gone by. Electronic mail has become common place. Furthermore, it has become easier and easier to communicate complex audio and video information via such computer networks.

A block diagram representation of a well known network configuration is illustrated in FIG. 1. In this well known configuration a network 100 includes a number of communication switches 101 to 104. The network represented by the cloud 100 provides points of ingress and egress for communications between users of the network. Included in the category of users are those service providers which seek to provide information or services to other users. For example, in an Internet environment various service providers may support different sites having different types of information. One site may have information regarding books, e.g., amazon.com, another might have information about news, e.g., cnn.com, etc. The service providers may have differing levels of sophistication in terms of the types of information that they market. Additionally, various service providers may have differing levels of needs with regard to authenticating users of the service and monitoring usage of the service for either billing or other informational purposes.

In today's Internet protocol (IP)-based networks there is efficient packet delivery that uses only the Domain Name Service (DNS) and routing for an in-network service. All the richness and diversity of services and applications is completely encapsulated in the client (user) and server end points. In some respects then the network is utilized simply as a transport mechanism with intelligence being pushed out to the outer limits of the network, that is, to the terminal points of the network where the users and service providers reside. There are significant costs associated with this arrangement, however, as more and more complex and sophisticated capabilities must be provided at the end points for each individual client or server. For example, in the client server configuration illustrated in FIG. 1 each individual server that wishes to keep track of its users and possibly limit access to only those who have properly registered with the service must have their own

separate capabilities for registering users and monitoring usage. Each server also should have then separate capabilities for authenticating previously registered users. Also, when appropriate the server should be able to effect billing. This complexity and cost can result in limiting the development of new services as fewer and fewer potential service providers are able to afford that which is necessary to perform even the most basic functionality need to track users.

It would be beneficial if an arrangement could be provided in which network end points would retain some intelligence to enable specific services which could become more and more sophisticated while a central resource could be shared by various ones of the service providers to perform common user tracking tasks that play roles in servicing using requests.

Summary of the Invention

The present invention provides a method and apparatus by which certain of the common user tracking functionality is off-loaded from the user/server network end points and is instead incorporated in the network architecture. As a consequence, the network assumes responsibility for such common user tracking functions as registration, authentication, user usage, and billing. In this arrangement a user who desires to register with services from two or more independent service providers can do so with a single registration operation involving core network functionality. The network operates as the receiving point for all user registrations and service registrations. The network can then maintain account records which represent access privilege information. This information defines the relationships between users and services with which they are registered. Once the user is registered with a number of services, if the user then at some later time requests network authentication and becomes active in the network, then the access privileges defined by the accounts information defines those services that the user will be allowed to access. Furthermore, all of the registration information can be made available in varying degrees of specificity to users and service providers alike. For example, users may be able to find out information about service providers and services that they render as registered with the network. Similarly, service providers

may be able to find out some set of information, limited by privacy concerns, regarding users or potential users of the provider's service.

The present invention provides this single registration capability in the context of a network employing a secure boundary with a plurality of access points or gates at which user registration and authentication are initiated. Core information maintained within the secured boundary is utilized to define user and service access privileges. Other features of the network will be described below.

Brief Description of the Drawings

FIG. 1 shows a block diagram representation of a prior art network architecture.

FIG. 2 shows a block diagram representation of a network architecture in accordance with an embodiment of the present invention.

FIG. 3 shows a block diagram of another version of the network architecture in accordance with an embodiment of the present invention.

FIG. 4 shows in block diagram form, examples of elements of a portion of the network architecture of FIG. 2.

FIG. 5 shows a schematic representation of an example of a multiple network configuration in accordance with an embodiment of the present invention.

FIG. 6 shows a schematic diagram of a logical control architecture for a network in accordance with an embodiment of the present invention.

FIG. 7 illustrates in block diagram form one aspect of an active register component of a network architecture in accordance with the embodiment of FIG. 2.

FIG. 8 illustrates a block diagram representation of a portion of a gate component of the network architecture of FIG. 2.

FIG. 9 provides a flow chart representation of a process for user access to a network service in accordance with an embodiment of the present invention.

FIG. 10 provides a block diagram representation of architecture components for implementing a usage and billing function in accordance with an embodiment of the present invention.

FIG. 11 provides a block diagram representation of the architecture of a peer

interface in accordance with an embodiment of the present invention.

FIG. 12 illustrates in block diagram form software components of the peer interface of FIG. 11.

FIGS. 13 to 15 provide flow charts for describing communication processing in the peer interface of FIG. 11.

FIGS. 16 and 17 provide block diagrams of network architecture components for enabling encryption to and from network access points such as the gates of FIG. 2.

FIG. 18 illustrates in block diagram form an embodiment of a gate employable in the architecture of FIG. 2.

FIG. 19 illustrates in block diagram form a network architecture that employs a plurality of gates of the type illustrated in FIG. 18.

FIGS. 20 and 21 illustrate in block diagram form data communication arrangements available using the network architecture of FIG. 19.

Detailed Description

Overview

FIG. 2 illustrates an example of a network in accordance with an embodiment of the present invention. In this arrangement network elements include a plurality of gates 201 through 206, a core, 207, a store, 208, a hop, 209, client peers 212 and 214, and a service peer, 216. The client peers represent those user or client entities that plan to interface with the network. The terms user and client will be used interchangeably throughout this document. The service peers represent those service provider entities that plan to interface with the network. Each of the peers is modified to include interface to the network. This interface can be constituted by software designed to run on the client or service provider computer equipment whereby the software interacts with interface software residing in the network gate elements. In accordance with the present invention the plurality of gates 201 to 206 form a secure boundary for the network such that only authorized clients and service providers can gain access into the network via their interaction

with a given gate at the boundary of the network. The store element 208 and the core elements 207 are considered part of the network capabilities and external peers such as clients and service providers are not permitted direct access to the stores and core elements.

The present invention incorporates certain functionality in the core and store elements such that functions that were provided by the service providers in the prior art configuration of FIG. 1 are now incorporated into the fabric of the network itself.

As an example of one function which might be incorporated within the network, the core can receive information from a gate with whom a client peer is in communication so as to receive user identification information. The core can then determine whether or not that user has already registered with the network. If the user has not previously registered with the network then the core and gate together can initiate a registration process by which the user registers with the network. At the time of registration the user can be provided with various options for subscribing to services that are presently available through the network. The various services can be provided by independent service providers. That is, the network can provide an opportunity for a single user to subscribe to or register with a plurality of services offered by different and independent service providers who are also at various end points of the network. All of this registration information can be compiled in account records or files in the core. When a user subsequently logs onto the network then seeks authentication and becomes an active user of the network, the network can then control the user's access privileges utilizing the account record information available in the core. For example, if the user has previously subscribed to service "A" but has not subscribed to service "B" then the network at the time of authentication can generate access privileges which permit the user to access service "A" while denying access to service "B".

In addition to the notion of maintaining access privileges for users and services the core and store elements of the present invention provide for centralized monitoring of a user's usage of the different services registered with the network. Pointing again to an example, assume that the user is registered with the network and has subscribed to services "A" and "B". As the user avails themselves of the

respective services, the network, rather than the services themselves, can monitor such usage and can even generate billing records should billing be an issue with respect to any one of these services.

The architecture of the present invention therefore provides a network in which there is a secure boundary which allows for secure transmissions within the network itself while the network incorporates certain functionality which would otherwise be common to a plurality of end users or service providers to reduce the need for complex processing at the end points of the network. The end points need not individually incorporate functionality which might be better shared between a plurality of service providers even if those service providers are independent of one another.

The following subsections of this detailed description will provide more information with regard to the architecture of this network and the specific elements which make up that architecture.

The Network Architecture

As described above in relationship to the schematic diagram of FIG. 2, the network architecture of the present invention utilizes five categories of hardware related components referred to as the core, gates, hops, peers and stores.

The remainder of this section will provide a brief description of each of these hardware components, their interrelationship with one another and how they are generally configured to establish the desired network architecture.

A core such as that shown as element 207 in FIG. 2 is the information kernel and heart of the network architecture. It maintains the business logic described as a rich web of relationships among logical entities that use the network for example, users, accounts, and services. Conceptually, the core forms an abstract layer that models the services via a set of objects and a well-defined set of relationships between those objects. The core contains information regarding user accounts and available services to support user authentication and access control. The authentication information contains cryptographic data that network security needs to authenticate a user's identity and to establish and maintain a session's security for

the duration of a user's connection to the network via a gate.

The gates, (201 to 206) represent the work horses of the architecture. They provide scalability, create a security perimeter, and enable protocol mediation. Gates implement a variety of security and service functions. These include access control supported by a dynamic firewall protection tightly coupled with the registration and authentication system, data proxies for various protocols such as Web content caching, and usage record generation for Peers. All traffic between the peers and the network flows through the gates.

Each gate contains two or more network interface cards, (NIC). One connects the gate to the private secure network using a packet filter as a switch and the other NIC connects the gate to the external unsecured network. Peers, stores, hops and other networks are on the external side of a gate and the other gates and the core are on the internal side.

Each peer (212, 214, and 216) utilizes its respective gate as an access into the network. The gate performs computing and administrative functions. The single gate supports access control, usage recording, peer control, and protocol mediation for its cluster of authenticated peers. A sample of a block diagram of a gate is illustrated in FIG. 4 which provides more details with regard to certain elements shown in FIG. 2. As can be seen, a single gate includes a packet filter 401, an Access Daemon, a Control Daemon 403, a Usage Daemon 402, a Data Daemon 403, a subsystem monitor 404 and a Unix Management Agent (MA) 405. Within the gate the access control consists of the packet filter and the Access Daemon that manipulates the access rules in the packet filter. Peer control is supported by a multi-threaded Control Daemon that spawns a separate thread for authenticated peer. Usage recording consists of the Usage Daemon that generates and locally saves the records and the usage proxy that forms the extended interface to the Usage Daemon. The Data Daemon runs separate threads for various proxies for protocols, such as HTTP, FTP, POP3, SMTP, Goffer, LDAP, H.323 and Telnet. Specialized proxies add additional support for functions like registration. The Daemons running on the gate support a centralized log-in, event notification, monitoring, and management interfaces.

Hops are the transfer points to other networks such as the Internet, the telephone switch network, or other enterprise networks. Hops also convert the IP traffic from the network gate to whatever is on the other side, such as Net Ware or PSTN. For telephony support, a hop consists of a H.323 gateway to the PSTN.

The network enables end-point devices in the positions that clients and servers normally occupy. These include personal computers (PCs), work stations, PBAs, set top boxes, telephones, cellular phones, or digital pagers. A peer can be any device that can be assigned an IP address and that implements the control channel protocol of the present invention. A peer can serve as both the client for end user applications and the server for services being offered to the network.

The software is referred to as the peer interface software enables these end-point devices to access the network via one of the gates. Peer device, in general, refers to an enabled end-point device, be it a Windows 95/NT client PC, a UNIX server, a Java station, a NC-dedicated terminal, or a Web Phone consumer device. In this context the peer devices are characterized as programmable with Java having an IP address and can communicate with an IP-based network. Devices that do not fall into this characterization can still be connected to the network given that there is a peer that can communicate with the devices.

The peer can be implemented with a set of Java classes. The use of Java satisfies a number of the functional and design requirements pertaining to robustness, easy extensibility, allowing incremental software updates, and allowing it to be embedded into dedicated machines. The peer software contains a framework for interacting with a network cloud. A cloud constitutes an authentication trust and centralizes authentication, access control and security for one or more domains. It is composed of Sites that together define the security domain. A site is the basic integration unit composed of some combination of gate, hop, store and core machines. Its definition is based on the existence of a secure network over which the gate and core machines are connected. This network can be either a private secure network or virtual private network over an unsecured network such as the Internet. It may or may not have a core for administrating a domain. The site centralizes functions like network management, network control, performance traffic

monitoring, bandwidth allocation, broadcasting, and multi-casting. Sites can be connected in order to provide the same functionality among larger numbers of dislocated machines. Communication between sites must be over a dedicated connection or if the sites are connected over an unprotected network, then the network level encryption of the link must be established.

With clouds it is possible to establish a partnership relationship, set up outsourcing, establish platform franchise or sell services. A cloud may have more than one core thus defining more than one domain. Each domain may be specified by a different domain ID and have its own registration and account repository. Users and services access the domain through any gate on the cloud thus logically sharing resources.

Returning to the notion of a peer, it can be executed in one of two ways, depending on the applications using it. For example, a Java virtual machine, (JVM), can be started and the main method of the peer class invoked. Alternatively, an instance of the peer class can be started from within an already-running JVM.

Applications that make use of the peer fall into two architectural categories. In the first, an application can be written in Java and run in the same JVM as the peer. This is the most natural mechanism for new applications and, in fact, may be the only possibility in certain environments, such as for embedded devices. Alternatively, an application can be written in a different language and communicate with the peer API using a socket-based protocol. A variation of the scheme would be possible in cases where a language an invoke Java methods directly (for example, Code can invoke Java using JDK1.1).

Given that some applications run outside the JVM containing the peer support and that it may be necessary to externally interface with the peer, an external interface called peer interface is provided. This interface accepts local connections and executes a limited number of commands using a simple protocol. A command-line interface utility called "pido" makes it easy to use this interface.

The peer also runs a mini HTTP server that lets browsers trigger actions in the peer. It is extensible by allowing developers to create new Java applications and register them with the peer in a way that associates a URN with the application. The

browser can then invoke the application with arbitrary arguments. The convention for extending mini-HTTP server with new CGI-like functions is currently not based on Java serveletts, but since serveletts are becoming ubiquitous and standard, using them will be a natural evolution of this design. Further details regarding the peer architecture will be provided below with regard to a description of FIGS. 11 to 17.

Returning to the notion of network clouds, although multi-domain clouds can provide logical separation and management of users and services, multiple clouds can be interconnected to offer a complete separation of domains. Clouds have to register with each other, much like services to allow traffic to flow between them. Once several clouds recognize each other, their users may be allowed to roam away from their home cloud. Roaming between clouds is the ability of a member of a given cloud to access the home cloud indirectly through other clouds. In the network of the present invention, roaming is supported by mechanisms that require a cloud providing roaming for members of other clouds to tunnel through to their other clouds. An example of this is illustrated in FIG. 5. In this example a peer 501 interfaces with a first cloud 505 in an attempt to gain access to its home cloud 510. In so doing, cloud 505 creates an encrypted tunnel 508 that cuts across that cloud and provides access to a gate on home cloud 510. Access control for the user is then performed at their home cloud. In this way, other clouds, such as 505, need not be relied upon to implement the user's access control, nor are they provided with any of the user's private information.

One network element that has not been touched upon as yet is the store. The store is a peer that implements services relating to maintenance, provisioning, and daily use of the cloud. This includes directories for white and yellow pages, customer care, recent usage records, download server, electronic mail, voice mail and paging messaging. In this regard there are also related databases such as customer contact and tracking, archival usage records, billing histories and historical network logs.

One additional feature regarding the network sites as described above is that each site maintains two domain name service servers. One is for the internal network and one is for the external network. The internal DNS is the slave to the

external DNS which is needed for resolving hosts names on the external network when requested from the inside. Typically, the external DNS runs on the gate that is connected to the Internet or the wide area network. The internal DNS typically runs on one of the core machines.

The network of the present invention has a physical architecture and a logical architecture. The physical architecture describes the interworkings of the computer and networking hardware and their connectivity. The logical architecture defines the relationship of the various components as they may be viewed by a customer. These relationships can be set to create various registration, security, access and authentication rules. The logical arrangement can be defined to meet specific business needs and environments. In this context the network logical architecture forms an entity called a cloud that can be constituted by one or more sites.

Having described the various components of the network architecture attention is drawn to FIG. 3 which illustrates another potential arrangement of network components to effect the goals of the present invention. Here a network site 301 includes a plurality of gates 305, 306, 307, 308, two core machines 312 and 314, a store 316 and a hop 318. These elements then provide access to external elements. For example, gate 306 is connected to peer 320 while hop 318 is connected to peer 325. Furthermore, gate 305 provides access to other networks. Similarly, the Site 301 can be coupled to another Site which includes two gates 330 and 335, a core 337 and a store 340. Information can then be exchanged between these network Sites. Alternatively, individual sites can operate in relationship to the peers connected thereto.

Platform Management and Monitoring

The network software components are managed and monitored by a distributed hierarchical system of monitors, sub-monitors and management agents. This is called the network management and monitoring system (NMMS). The networking and hardware components are managed using an SNMP-based third party Enterprise Management System. The combined use of these two systems provides a comprehensive management and monitoring capability for the network.

The NMMS can be a three-layer hierarchy consisting of a system monitor (SM) running on one of the core machines, a single subsystem monitor (SSM) running on each machine to be managed, and a Management Agent linked into every Daemon to be controlled. This hierarchy is shown in FIG. 6. The SM and SSM processes are started (and restarted) automatically buy a Unix Init mechanism. Upon starting, the SM and SSMs process broadcasting an announcement of its presence to locate each other. The SSM are responsible for monitoring the health and starting the various local Daemons under their control. The management agent interface is designed to run even if its SSM decipher is not present. The NNMS can be brought up or taken down in its entirety without affecting the operation of the network system.

The system monitor is the nerve center of the system. It directly controls and communicates with the subsystem monitors. The subsystem monitors are the local control monitors. These system monitors and subsystem monitors are illustrated in FIG. 4 in the gates which include the subsystem monitor and the core which includes the system monitor. The subsystem monitors channel events between the system monitor and the Daemons and monitor the health of the Daemons. The system monitor also exposes a CGI-based connector for communicating with a browser-based administrator interface and a command line interface (SMCLI) also shown in FIG. 4. The browser downloads Java applets from a web server running on a core machine. These Java applets interface with the CGI interface in the system monitor. NMMS is registered as a private service. Administrators authenticate from any peer just as any users, establish an encryptic connection and use the browser to access NMMS.

The SM, SMMs, and MA-enabled Daemons form a communication network for delivering commands from administrators' interfaces, and returning responses from the controlled Daemons. However, not all Daemons can be extended with a MA interface. As such, Daemons can be controlled in one of two ways; either directly through a linked MA library or externally through an UnixMA process which contains the MA library. Typically, there is one UnixMA process associated with each SSM. Using UnixMA, the administrator can check process status, read

data files, or do most anything that could be done sitting at the machine's console. UnixMAs are also responsible for polling. When SM starts, it starts a Poll Keeper 460 (see FIG. 4). The Poll Keeper reads the configuration of the system and starts a poller in the appropriate UnixMA process for each component or function to be polled.

Each component comes standard with five command-line scripts. These can be used manually by an administrator when logged into the machine. These commands are:

start_component. Starts the component.

stop_component. Shuts down and stops the component.

describe_component. Prints out a full description of the component including its version number and indication of its installation status.

status_component. A short version of the full description provided by the script describe_component.

running_component. Prints out the components running status.

However, the benefits of management and monitoring lie in the direct use of MA capabilities. MAs can control the logging level of their components, obtain CPU usage, memory usage, file descriptor usage, start, stop and reinitialize components, obtain the version numbers and generate alerts. MA-enabled components can also register new commands that can be invoked by the administrator to provide customized control of internal functions.

NMMS maintains the description of the entire system. This information comes partly preconfigured and is partly collected during an interrogation step prior to installation. The network architecture can be perceived as a collection of resources (objects) which communicate with each other, share properties, can be created and destroyed. Each resource has associated with it a description, given as a collection of attributes. The objects describing the resources are maintained by a component called SysReg (as shown in FIG. 4). SysReg consists of a database of name/value pairs and a front-end server that accesses the database. Supporting application program interfaces (APIs), for accessing the server, may be provided in several programming languages.

Network components that are installed, configured, and managed by NMMS are implemented on top of three support libraries, Log, Ma, and MACON.

Log is a C API that includes assert-type macros, trace macro, and C functions. A Daemon generates log file entries by using assert and trace macros. Whether log file entries are generated depends on either compile time or run-time parameters. At run time, three parameters control the logging and tracing levels. These can be set to one of the values **none, error, warning, notice, info,** or **debug.**

assertLevel controls the level of assert macros. In addition, **alertLevel** determines if the macro generates an alert.

logLevel controls the level of log macros. Here, also, **tracealertLevel** determines if the macro generates an alert.

traceLevel controls the trace level.

One additional parameter, **panicThreshold,** specifies the number of entries after which message generation should stop. This is a safety valve in case logging is left on unattended. Initial values of the Log library parameters can be specified through environment variables. They can then be obtained and modified at run time through C function APIs.

MA library provides an interface between network Daemons and NMMS. Simply by linking with MA, a Daemon enables the monitoring system to access its internals. MA library provides a number of built-in commands. In addition, every Daemon can define and export to NMMS any command relevant to its monitoring and management functions that will make use of the monitoring commands exported by Daemons.

MACON (Management Agent CONsole) is a program provided to the Daemon developers for interactive invocation of the MA commands in their Daemons during development. It plays the role of a SSM (SubSystem Monitor) in a deployed system. The developer using MACON plays the role of a SM (System Monitor).

Network Component Functionality

The network of the present invention provides certain functionality

including, but not limited to, membership control, tracking registered users and services, access control and usage recording. This subsection will describe various features of these functions.

A membership subsystem provides support for registering users, services and other clouds. The subsystem also supports profile updates and information retrieval regarding such user services and clouds. Registration information is kept in an account hierarchy that can consist of accounts as the internal nodes of the hierarchy with users, services and clouds as the leaves of the hierarchy. The relationship between the parent and siblings in the hierarchy dictates access rights and privileges.

An account in this membership subsystem can be considered a collection of users who belong to the account and those services that own the account. Each account may have the following fields and attributes:

1. identity--each account has an identity, (that is, it has a registered name and an id number);
2. parent account--all accounts, except the root have parent accounts;
3. list of users--list of users who belong to this account;
4. administrators--some users are designated to be account administrators, they can create (add) users to the account and create subaccounts;
5. list of services--list of services that are owned by the account, the services can be stopped and started by the account administrator;
6. list of subaccounts--list of children accounts;
7. list of privileges--set of services that may be accessed by the account users.

A user is a person whose identity is known to the cloud and can be authenticated by the cloud. Each user has the following fields:

1. identity--each user has an identity (that is, it has a registered name and an ID number);
2. membership--each user belongs to one account; and
3. role--account administrator or regular user.

Any registered service running on a peer whose identity, location, and access method are known to the network are included as a service in one or more accounts. A service has the following fields:

1. identity-- each service has an identity (each service has a registered name and an id number);

2. ownership-- each service is owned by an account, it can be started and stopped by the account administrator;

3. support services-- a service can rely on other services; and

4. subscription list--for each service there is a list of users who may access it, at an abstract level of services at the minimum just a set (group) of users.

In order to avoid recording privileges for all services, and in order to simplify evaluation of privilege and access rules, the notions of public and private services may be introduced. Public services are those that can be accessed by any user, unless the user is explicitly blocked from accessing the service. Private services are those services that cannot be accessed simply by any user, but instead require that the user be explicitly granted access. To access a private service, a user must first subscribe to that service. Private services can be of two types or two subscription policy types, a public subscription or private subscription. In a public subscription policy any user is permitted to subscribe to a service. In a private subscription policy limitations are placed on the user before they are allowed to subscribe to the service. For example, certain additional requirements may be set before the user is permitted to subscribe to the service.

The distinction between the public service and the private service that has a public subscription policy is that the user must subscribe to a private service before they are allowed to access it. This is an important difference as the act of subscribing to a service can cause a usage record to be generated as will be described below and that event has the potential to be transformed into a bill to the user.

Private services with private subscription policies provide even tighter control over who may subscribe to the service. These subscription policies are implemented by associating a subscription service with the private service. A subscription service provides a mechanism to screen users prior to subscription. An API is provided whereby the subscription service can permit or deny users from subscribing to the private service. By running a subscription service a service operator can enforce any arbitrary policy over what users can subscribe to their

service.

Only private services with private subscription policies need subscription services. Services with public subscription policies allow all users to subscribe and therefore do not have subscription services associated with them. In either way, these subscription services can be implemented within the network of the present invention, that is they can be considered functionality that is absorbed within the architecture of the network removing it from the service end points where it would otherwise reside in prior network configurations. This would involve keeping track of the characteristics of those services which are registered with the network and implementing subscription APIs within the network to solicit information from a user (for example, user identification information and billing information) to set up the subscription. The network could then keep track of the subscription and report subscription results including billing information and other user profile information to a given service.

In addition, the network of the present invention may provide an automated approval system whereby the user is given approval by its parent account to subscribe to the service. Since the approval services and the subscription services can be operated as APIs with individual functionality recited for those services, it is not necessary to substantially affect the operation of the core to implement arbitrary subscription policies. Nonetheless, the network configuration permits the individual services, having described specific subscription policies, to perform subscription and approval operations without need for maintaining complex equipment at the server end point.

The network architecture of the present invention also provides a technique for maintaining information about those users and services which are active with regard to the network. In that regard, an active registry such as that shown in FIG. 7 is a component of the network core. This active registry (AR) 701 serves as a maintainer and supplier of information about all currently authenticated users and services. Thus, AR can be thought of as a dynamic directory or an Internet Locator. As user and services log-in (authenticate), information about them and their current location is added to the registry. This information is supplied on demand to other

components, such as for access control decisions. The information is also sent to the White Pages over an LPAP interface. This allows a more detailed search by people to locate active users and services. For currently authenticated users, (also referred to as active users), the registry contains: a user identifier; a user handle; an IP address of the peer from which the user is currently logged in; and the home proxy of the user (that is, the gate to which the peer is directly connected), Active User Registry (AUR) 702. For currently authenticated services, the active registry contains: a service identifier; an IP address of the service peer hosting the service; an IP port of the service; and an IP protocol used by the service, Active Service Registry (ASR) 703. The information in the AUR 702 is used to implement Caller ID and intercloud decisions. It is also available to client peers for services such as telephony and collaboration. The AUR 702 runs within the AR 701 as a thread that listens on port 2221 for requests from control Daemons 720 of a gate 730. Likewise, ASR 703 listens on port 2223. The protocol used to communicate with the AUR and ASR is based on the HTTP\1.0 GET method. This protocol is hidden behind the application program interfaces. Two different interfaces implement the protocol to communicate with the AUR and ASR. Each exposes a subset of the functionality of the AR that is needed by the target audience of the API. From peer applications, AUR can be queried for the IP address of an active user. The control Daemon communicates with the AUR to add a user, such as peer 760 on logon, remove users at logoff, and query for active users based on an IP-address and user handle. The same is true for the ASR. A program could be supplied with the AUR to help with the administration. Such a program could provide the options to: dump all entries in the AR; search for an entry in the AR by either the IP address or by user or service handles; delete an entry from the AR; or add an entry to the AR.

As a result of this component of the core the system provides the capability for maintaining updated information identifying those users and services registered with the network, that is active on the network, and their respective locations with regard to the network (such as an IP address and information about the gate to which it is connected).

The network also has an access control system that allows or restricts access

between users, services and other networks. This ties in with the membership control system in that account information or records which are created as users and services register and subscribe in connections to the network are used to impose control restrictions during access attempts by the users or services. The access control system provides mechanisms that allow data proxies to impose access control restriction by mediating protocols used to communicate between users and services. Access control is enforced collectively in a gate in the packet filter, the Access Control Daemon and the Data Daemon. An example of elements contained within the gate which effect this access control are illustrated in FIG. 8. It should be understood that all of the elements of the gate are not illustrated (for example reference to FIG. 4 indicates that additional elements may be contained in the gates), but the elements illustrated in FIG. 8 are instrumental in providing access control.

In accordance with the control mechanism traffic is only allowed to flow through the network if it passes the access control test. The smallest granularity that is provided by the access control is that of a single user or a single service. As an example, a particular user can be prevented from accessing a specific page on a particular web site. This could be handled on a protocol by protocol basis in the data proxies 805.

In some cases access control needs to be enforced upon users and services that are not running on peers. A news feed is an example where this is necessary. To accomplish this these users and services are registered using the standard registration procedure. Instead of authenticating them by running peer software they are preauthenticated when the core network starts. The active user registry can then be prepopulated from the pre-authenticated user list. The active service registry is prepopulated from the preauthenticated service list. From the perspective of the access control subsystem they are treated the same as any other user or service. During registration services can also be designated as accessible by guests. Such designated services allow guests (non-authenticated) users to access the service.

Initial access control data is generated at user registration time. The users are presented with a list of private services to which they may subscribe. The subscription operation is described above with regard to the membership subsystem.

It should be noted that users can also subscribe to services after registration. For example, a registration update procedure could be provided whereby a user, upon authentication is given the option of subscribing to an additional service or services. Alternatively, the registration system could require that an authenticated user take some proactive step to contact a subscription service such as described above.

Returning to the notion of how the access control is effected, the packet filter 801 in FIG. 8 is the first component to enforce access control policies of the access control system. It receives all incoming network traffic. The packet filter is rules based. Whenever a packet arrives on the external network interface, the filter analyzes it and, if it is not part of an existing session the rules, Access Rules 820, are consulted to determine what action to take. The rules are generated on a demand driven basis. Initially, there are no rules loaded into the filter.

When a packet arrives for which there is no matching rule the filter executes a check action and passes the packet up to the Access Control Daemon 840 for further analysis. The Access Control Daemon performs the access check and installs an appropriate rule into the filter that covers the attempted action. The new rule may contain a DROP action if the traffic is not allowed; it may contain a PASS action if the traffic is allowed; or it may contain a LOCAL action if the traffic is to be sent to a local proxy. After the new rule is installed into the packet filter the packet will be returned to the filter for reevaluation. The filter will then trigger the new rule causing the appropriate action to be taken. Once the new rule is in place, subsequent access between the same user and the service will trigger this rule thus bypassing the Access Control Daemon. In the case where the traffic is destined for a local proxy the Access Control Daemon may not be able to determine the actual service being accessed. HTTP is an example where the destination is embedded in the data stream. For these cases, an additional access control test is performed in the data proxy. As can be seen from the architecture represented in FIG. 8, the Access Control Daemon relies on the ability to communicate with a registration server (not shown) and the active registry so as to be aware of those users and services which are active within the network and also to access account records which indicate which access privileges a user or service may have.

FIG. 9 show a flow chart illustrating a process for access control in accordance with a method of the present invention. In this example the access control is operated on behalf of a user seeking access to a service. The same process can be applied to a service provider or server seeking access to a user or to user information. In this example, the gate receives an authentication request, step 900. The gate then can execute an authentication proxy, step 901. This step of authentication determines whether the user who seeks access to the network is one who has previously registered with the network and is a valid network user or service. The system determines whether the user is authenticated in step 903. If it is not, then the connection is dropped, step 904. If the user is authenticated, then the user sends a service access request to the gate. The service access request is received in step 905. The gate determines whether a filter rule is available. As described above, a filter rule could already be preloaded in the packet filter. Alternatively, the Access Control Daemon may have a rule available which could be loaded into the packet filter. If it is determined that the gate does not have a filter rule available for this access request then a rule can be retrieved, step 907. This rule can be retrieved in accordance with access control privilege available in accounts records at the registration server taken in conjunction with active registry information available in the core. If the gate has an available filter rule, or it has retrieved such a rule, then the gate applies the filter rule to the service access request, step 907 thereby either passing a request into the network or dropping the request.

Various access control techniques can be employed in the gate taking advantage of specially designed proxies and the packet filter which can act as a firewall. More specific examples of such techniques are described in "System and Method for Demand-Driven Loading of Rules in a Firewall", "High Resolution Access Control", "System and Method for Distributed Access Control with a Centralized Database", "Access Control for Applications with Dynamic Parameters", and "System and Method for Network Loading Balancing".

As described in the subsection "Supergate" below a modification to the gate architecture is envisioned in which the gate will have the capability of not only determining whether to route a user into the network based on certain filtering rules

but will also have the capability of routing a user along the periphery of the network without unnecessarily tying up too much of the network's resources.

Having described how the user can become registered with the network and how its access control privileges can be set and utilized we turn our attention to another network function which, in the prior art, was assumed by end point servers, this being usage recordation. In accordance with the present invention the store of FIG. 2 may incorporate a usage recording server. An example of this is shown in FIG. 10 of the present application. A usage recording and retrieval subsystem provides usage generation, collection and storage and retrieval. This subsystem records usage-related events for purposes of billing, custom care and network usage analyses. The usage recording and retrieval subsystem includes Usage Daemon 1010 and collects usage records from processes running on gates such as the Control Daemons, Cache Daemons, and the data Daemons. It also collects records from processes running on service peers, through a usage proxy 1012. This subsystem also retrieves usage records from a usage database 1040 for the service peer based on its request for users wanting to inspect their record.

A usage record may have multiple fields. In one embodiment a usage record has the following fields:

1. ID of user's home cloud;
2. ID of the gate where the record was collected;
3. time of event recording on the gate;
4. record sequence number;
5. record type name (session, content,...);
6. record subtype, destination cloud ID, service ID;
7. ID of user responsible for record generation;
8. unique ID of session during which record was generated;
9. time of usage event; and
10. even description-name/value pairs.

Each usage record has a minimum length, for example 76 bytes, plus the variable length of field. Four types of usage records may be employed: session records, content records, intercloud records, and service generated records. Session

records are generated by a gate's control Daemon when a user of a service logs on, logs off, and for periodic heartbeats recorded during the session. Content records are generated by the gate's data proxies for HTTP, FTP, E-E-mail and other content protocol transfers. Intercloud records log IDs of destination clouds. Service Generated records log the service IDS registered for usage recording.

The gates collect all usage via the usage Daemon 1010 and the Usage Proxy 1012. Again, FIG. 10 represents elements which may be contained in the gate although these are not an exhaustive representation of the elements which may be part of the gate.

The Usage proxy 1012 may be implemented in C++ as a shared library. The Usage Proxy accepts requests from Peer Usage libraries to either drop or retrieve usage records. The Usage Proxy communicates with a Peer over a Usage Retrieval Communications channel. It enforces access control on all operations, but it does not check the validity or correctness of usage records that it forwards to the Usage Daemon.

Usage Daemon 1010 performs the work of validating the correctness of usage records being dropped and interacts with the Usage Server process on the Usage Recording Server machine 1060. It is controlled by the following four parameters:

Max Push Interval Time - The time interval after which a file of usage records is prepared, providing there is something to prepare.

Dispatch Interval Time - The time interval to move the files containing the usage records to the directory monitored by the Usage Pusher.

Polling Interval Time - The time interval for Usage Daemon to wait in the polling queue for data to arrive from the various usage generators.

Usage File Watermark - the maximum size of the Usage File that can be created within Max Push Interval Time.

Usage Pusher Daemon (1030) - Keeps checking the directory on the gate receiving all usage records from the Usage Daemon and, when appropriate, flushes them to the Reducer (1055) in the Usage Recording Server. The data is encrypted via the standard Gate/Peer crypto system described below.

Usage Recording Server can reside in any Store. It depends on a relational database system such as Oracle, an HTTP server such as Apache, and a Reducer.

Reducer 1055 receives the usage data from all the Usage Pusher programs in the gates. It formats and stores the usage records into the Usage Database. It performs some data reduction depending on the type/subtype of the records. In particular, it performs matching of the user's log on and log off records.

Usage database 1040 accepts and stores the usage records processed by the reducer. It is organized by the type and subtype of the record. For each type/subtype of a record and for each service generating records there is a separate table in the database. This database organization facilitates implementation of selective usage record handling policies. The database also contains the data retrieval information identifying what services have access rights to the usage records. The access rights are specified on a per-usage record type/subtype basis. In other words, for each type/subtype of the records the Usage Database contains explicit information about services that can request their retrieval.

HTTP Server 1070 and HTTP Wedge 1080 provide usage feedback to the user through a browser interface. The HTTP Wedge 1080 provides caller ID verification to the CGI programs retrieving the user's usage data from the Usage Database 1040. The ownership of the usage data is interpreted as follows:

each user can view only their data;

account administrators can view the usage data of all users associated with the account and the subaccounts;

the service administrator can view all usage records generated by the service.

The functionality described permits the network to assume the responsibility for various activities that previously resided in the end points of the network. The network provides the necessary resources for such functionality as authentication, subscription, access control, usage recordation and billing thereby taking the burden off of the end points of the network and bringing that functionality within the secured boundaries of the network. As described earlier, this reduces the complexity and cost of the components necessary at the end points. Other functionality may also be provided in connection with this network architecture. For example, the network

can provide for a general mediation of protocols. A network can easily manage and manipulate all standard protocols to add values as necessary. For example, the network can transparently support a distributed cache architecture that intelligently monitors and caches documents flowing over the web which uses standard protocols. This is but one example of protocol mediation which could be obtained with the present invention. In addition, the network architecture also provides for information integrity by facilitating data stream IP layer encryption/decryption between the gates and user or client peers. This encryption technique will be described below in connection with the peer software under the subheading Peer Interface.

The network architecture set forth above therefore provides a network with a secure boundary that incorporates within that boundary certain intelligence and certain common processes which can be shared by a plurality of independent users and a plurality of independent service providers. The architecture provides that a user can access various network services using a single registration or authentication process. Once logged-in, the network access controls take over, referring to account record information, to permit it access to multiple, independent services.

Peer Interface

The functionality of the peer interface has been described above in the subsection regarding Network Architecture.

An example of potential modules which could form the basis of the peer architecture are illustrated together in FIG. 11 of the present application. There it is shown that there are network aware applications as well as non-network aware applications. The peer interface itself includes ISP modules, application support modules and other application program interfaces. Furthermore, the peer includes a socket redirector. The peer software can reside in a directory on the peer identified by the value of an environment variable. This directory contains all executables, job of classes, applications, and property files used by the peer. Every application has its own directory in their ID/apps/ in which it can do whatever it wants. The bin/ directory contains the executables. The lid/ directory contains miscellaneous data

files. The classes/directory contains job of class files. The lid/jars/ directory contains Java files for updating and installation. The Ap.properties and user.properties are two important files found in the lid/ directory used for customizing the peer and configuring user preferences and for retaining the user's identity. The app.properties file contains all peer, network and applications-specific properties. The user.properties file contains whatever is necessary for a user to transport their identity to another peer machine.

From the moment that the peer attempts to connect to the network, a single control channel gets established between the peer and the Control Daemon on the gate. This control channel is the tether that binds a peer to a particular cloud and maintains an active session with the cloud.

The peer software resides on the end points that connect to the network. The software facilitates making secure connections to the cloud and makes use of registration, authentication and other services provided by the network. The software may be run on such operating systems as Microsoft windows '95, Microsoft Windows NT4.0, Fun Solaris, SPARC, Linux i 86, and SGI IRIX platforms. One of the integral components of the peer software is the redirector. The redirector module facilitates the capability of transparently encrypting all IP communication from the peer to the gate and vice-versa. The redirector makes use of platform dependent facilities (Winsoc 2 on Win32 and Streams on UNIX) to intercept IP communication without requiring any changes to the applications running on the platform. Thus, existing applications such as Netscape Navigator, Internet Explorer, Internet information server, etc. transparently inherit added security and privacy.

FIG. 12 illustrates the working of the redirector component. the decision module 1210 determines whether encryption should be enabled for the given socket connection. The encryption module 1220 performs encryption/decryption of the data that is transferred over the socket.

The peer uses a platform specific IPC mechanism such as shared memory to pass information to the redirector. This information includes: 1. a global encryption flag that specifies the current state of encryption (on/off) between the peer and the

gate; 2. the encryption key that gets used to encrypt/decrypt the data; 3. the lookup table that helps redirector to decide whether to encrypt a given socket connection.

The peer can initialize the look up table based on the configuration specified by the user and a few default entries. Each entry in the table describes a set of hosts using "network address", "netmask", and "port number". The entry also indicates whether the encryption should be enable for the hosts that match this description.

This information may be protected in the peer by a "single writer/multiple reader" synchronization lock. Thus, only one writer can access the information at a time and cannot access it while it is being read.

When an application tries to establish a socket connection, the redirector invokes the decision module. The decision module first looks at the global encryption flag. If this flag is "false" (off) the encryption is not performed. If this flag is "true" (on), the decision module compares the IP address and the port number of the host on the other end of the connection with each entry in the lookup table. The search is performed from the top to the bottom and stops as soon as a matching entry is found in the table. The encryption flag in this entry determines if that connection will be encrypted. The matching entry is not found in the lookup table, the default is to encrypt the connection. This default can be changed by adding an entry at the bottom of the table that would produce a match for any IP/port combination.

The flow charts at FIGS. 13 to 15 illustrate the flow of control in the redirector for various conditions.

FIG. 13 shows flow of control in the redirector when an application tries to establish a socket connection. After the information is locked so that it cannot be written while read, the socket redirector determines whether the encryption is on, step 1301. If not, then the lock is released and the connection can be completed. There will be no encryption in that circumstance. If, however, the global encryption flag is on, then the system determines whether the encryption table says that encryption in this circumstance should be carried out. If the answer is no, then again, the socket connection can be established without any encryption. If, however, the encryption table indicates that for this particular connection encryption is

required then the redirector reads the session key, step 1304 and creates an encryption for the given socket, step 1305. The lock is then released and the connection is completed using an encryption with a specific key for that particular socket. Naturally, the peer may be utilizing different sockets for different connections, some of which may be encrypted using various session keys while others may not be encrypted.

FIG. 14 provides a diagram that shows flow of control when an application tries to send data over the socket. The application sends the data to the encryption module of the redirector. If the encryptor for this socket is valid as determined in step 1402, then the data is encrypted, step 1403 and sent to the gate. If the encryptor for the socket is not valid then there will be no encryption and the data can be sent without the encryption.

FIG. 15 illustrates a circumstance where an application at a peer tries to receive data over the socket connection. The data is received by the peer, step 1501 and the peer redirector performs a real receive operation, step 1502. If the encryptor for the socket is valid as determined in step 1503 then the received data is decrypted in step 1504 and control is returned to the application running on the peer. If, however, the encryptor is not valid then the control is returned to the application without decrypting the data.

This encryption/decryption of the data stream IP layer between peers and gates provides information integrity. The encryption and decryption are handled differently on the peers than on the gates, however. On the gates encryption/decryption is done in data proxies, not in the packet filter. On the peers the encryption/decryption is not done in the applications, but at the transport layer by the socket redirector as described above. With this arrangement, all existing applications can safely communicate with the gate without change.

FIG. 16 provides a representation of how the data flows between a win 32 peer application and the gate. The flow is similar for a Solaris peer.

The socket redirectors, both Win 32 and Solaris, do encryption using RC 4 variable -key size stream cipher. They use the accesses Leay libraries RC 4 implementation with a key length of 128 bits. Of these 88 bits may be exposed

during key negotiations to satisfy export requirements. The encryption process works by exclusive ORing the bytes of the plain text with a stream of randomly generated numbers. This key is generated during the authentication process for this session and thrown away when the user logs off.

An example of a configuration in which the sender and receiver perform an encryption/decryption process is illustrates in FIG. 17. Here it is shown that the sender and receiver both include pseudo-number generators 1701 and 1702 receiving a shared secret key. The random number stream from the pseudo-number generator is then interacted with the plain text stream in the sender to create an encrypted stream. Similarly, the pseudo-number generator, influenced by the shared key, is used to decipher the text at the receiver.

In summary, the peer software may consist of a plurality of software modules which enable control of the communications between a user or server and the ingress or egress points of the network, that is, the gates. The modules operate so as to enable encryption for the communications between the peers and the gates without affecting operations at the application program layer in the peer machines. In essence, the encryption technique is substantially invisible to the application program employed in the peer machines.

Super Gates

In the general description of the network architecture provided above, one of the key components to providing a secure network boundary is the gate. The gate acts as an ingress/egress point for users or service providers as they interact with the network. One embodiment of a gate configuration utilizing various Daemon and proxies has been described above. The gate plays an important role in access control and in usage monitoring.

The gate would be improved if it could intelligently deal with situations where an active user desires information from an active service provider and the gate could intelligently decide whether to route the communications between these two end points either directly through the network or along a peripheral boundary of the network. This could be an important decision depending on the network resources

which might be consumed in effecting the information transfer. For example, a given user may desire to have access to a service that makes available a video stream. Transmission of that video stream back to the user would consume a tremendous amount of bandwidth and if routed through the network itself would consume network resources. It would be beneficial therefore if after authenticating a user and identifying that they have access control privileges to a given service a gate were to route information between the user and the client not directly through the network but through external interfaces which may be better equipped to handle the bandwidth requirements of the transfer or at a minimum may avoid overly burdening the network resources.

The network architecture of the present invention includes a modified gate structure which can be referred to in shorthand as a Distributed Network Element or super gate. An example of a super gate is illustrated in FIG. 18 of the present application. This super gate would include together with an ATM switch fabric represented by the Network element 1801 and gate control capabilities, represented by the Network Node 1802. An ATM network interface card for interfacing the CPU of the gate and the ATM switch fabric. This interface is shown as the Network Element Adaptation Layer, 1802. The super gate would thus be connected to an ATM network via the ATM switch fabric.

FIG. 19 illustrates an example of a network architecture that incorporates the supergate distributed network element configuration. Two examples of such elements are more clearly delineated by elements 1901 and 1902 although each pair of a gate (e.g., 1905a) and a switch (1905b) can be considered a supergate. As is clear from the figure the secure trusted domain boundary can be considered to extend out to the switches (1901b, 1902b, etc.).

The supergate structure and revised network architecture provide for flow separation. Small volume flows requiring special handling and control tasks can be routed through the gate into the network. Large volume flows, on the other hand, are routed through the switch controlled by the gate to the ATM network under control of the gate and can ultimately be routed to its destination either along the periphery or external to the network itself. An example of the separation of low and high

bandwidth communications is illustrated in FIG. 20. Again, the secure trusted boundary extends to the switches that make up the supergates (e.g., 2001 and 2002). Only two such structures are shown although the boundary could be constituted completely by such supergates. In this arrangement the gate portion of the distributed network element controls how the switch routes data after authorization or registration has occurred. In the illustrated example a first client peer 2030 may desire high bandwidth access to video server 2050 which is coupled to the network via super gate 2002. Client peer 2030 is coupled to the network at supergate 2001. At the same time client peer 2040, coupled to the same supergate 2001, may seek a low bandwidth communication to store 2035 which is also coupled to supergate 2002. The network node 1803 and network element adaptation layer 1802 control the network element so as to route the high bandwidth communications between supergates 2001 and 2002 for example along a path external to the network such as via Internet 2010. The network node 1803 and network element adaptation layer 1802 control the network element so as to route the low bandwidth communications between the supergates 2001 and 2002 through the secure network via the connection represented by path via 2020.

This architecture provides the advantage of allowing high volume flows to cut through an ATM switch controlled by the gate while still sending important low volume flows through the gate's protocol stack.

FIG. 21 illustrates a video streaming traffic path using the architecture illustrated in FIG. 19 and the supergates of FIG. 18. In this arrangement the video stream traverses the secured network between the gates of supergates 2101 and 2102 to achieve the desired secure data communication. The topology of the network shown in FIG. 19 may easily be altered to allow the support of different traffic patterns. For example, the ratio of total low-volume traffic (processed by full protocol stack at the gate) to high-volume cut through traffic may determine the number of ports used for gate-switch communication. Also, the full connectivity matrix may be replaced by a more sparse one which will reduce the cost and still maintain easy path to future upgrades. The number of gates, the gate capacity and cloud capacity can easily be incremented by adding more switch ports. The

existence of alternative paths also improves reliability and reduces call blocking probability. This distributed network elements architecture may be easily distributed to a distributed core architecture if the core is expected to become the bottleneck. The fact that the distributed core lies within a trusted domain may remove any implementation detail related to security.

The gate of the distributed network element or supergate can execute control of the associated ATM switch using the General Switch Management Protocol (GSMP). GSMP allows the following: establishing and releasing connection across the switch; adding and deleting leaves on a point to multi-point connection; managing switch ports; requesting configuration information ad statistics. Command line and web-based tools are developed that allows the user to set up rules at the IP-address port level and needed to define the quality of service or do security/filtering y dropping packets.

This arrangement of the super gate or distributed network element therefore provides an improvement in traffic flow whereby high volume traffic does not necessarily negatively impact the operation of the gate mechanisms thereby causing a choke point for other information that must be provided to the network.

Conclusion

The network architecture of the present application provides for reducing the cost and complexity of the end points by incorporating certain functionality within a secured network boundary. Furthermore, access to the network is governed by software control modules which can be loaded on end points and effect the transfer of secure information without adversely impacting upon the application programs running on the end user. Also, a gate architecture for the ingress and egress points for the secure boundary of the network provides flexibility by combining certain gate functionality along with control over switch so as to bifurcate the treatment of low volume and high volume data flows.

WHAT IS CLAIMED IS:

1. A method for facilitating communications between a plurality of users and a plurality of independent service providers, the method comprising the steps of:

receiving a registration request from a first user;

receiving a request from the first user to subscribe to a first service provided by a first one of the plurality of service providers;

storing account information identifying the first user and access privileges with regard to said first service;

receiving a registration request from a second user;

receiving a request from the second user to subscribe to a second service provided by a second one of the plurality of service providers; and

storing account information identifying the second user and access privileges with regard to said second service.

2. The method of claim 1 comprising the further steps of:

receiving a service access request from the first user;

determining whether the first user has access privileges to the service that corresponds to the service request, said step of determining being performed in relation to account information associated with the first user.

3. The method of claim 1 comprising the further steps of:

receiving a request from the first user to subscribe to a third service provided by a third one of the plurality of service providers; and

storing account information identifying the first user and access privileges with regard to said third service.

4. The method of claim 3 comprising the further steps of:

receiving a service access request from the first user;

determining whether the first user has access privileges to the service that corresponds to the service request, said step of determining being performed in

relation to account information associated with the first user.

5. A method for controlling user access to services via a network, the method comprising the steps of:

registering a plurality of services from a plurality of independent service providers;

storing account records identifying registered services and users permitted access to the registered services;

receiving a registration request from a first user;

authenticating said first user;

receiving a service access request from the first user; and

determining whether to connect the first user to the service associated with the service access request wherein a decision in said step of determining depends on information in the stored account records.

6. The method of claim 5 wherein said decision is based on an access filter rule created with reference to the stored account records.

7. The method of claim 6 comprising the further steps of:

after authenticating said first user, updating an active registry to reflect that said first user is active.

8. The method of claim 5 comprising the further steps of:

connecting the first user to the registered service; and

monitoring user usage of the requested service; and

generating usage information based on the results of the step of monitoring.

9. The method of claim 8 comprising the further steps of:

authenticating a second user;

receiving a service access request from the second user; and

determining, depending on information in the stored accounts records,

whether to connect the second user to the service associated with the service access request received from the second user;

wherein the service provider of the service requested by the second user is independent of the service provider of the service requested by the first user.

10. The method of claim 9 comprising the further steps of:

connecting the first user to the service requested by the first user;

connecting the second user to the service requested by the second user;

monitoring usage by the first user of the service requested by the first user; and

monitoring usage by the second user of the service requested by the second user.

11. A communication system for interfacing a plurality of independent users and a plurality of independent service servers, the system comprising:

a first peer interface associated with a first one of the users;

a first gate, coupled to said first peer interface, the gate including an access control filter that executes access control rules relating to said first one of the users;

a second peer interface associated with a first one of the servers;

a second gate coupled to said second peer interface; and

a network core coupled to said first gate and said second gate, wherein said core includes an authentication subsystem that creates information for the access control rules.

12. A method for controlling communications between a user and a service provider, the method comprising the steps of:

receiving, from a first user, an authentication request at a first gate;

authenticating said first user using said authentication request;

establishing an encryption protocol between the first user and said first gate;

receiving, from the first user, an encrypted service access request at said first gate;

decrypting said encrypted service access request;

determining whether said first user is authorized to access the service identified in the service access request, and if said first user is so authorized;

sending service related information across a network to a second gate,

encrypting the service related information at the second gate, and

sending the encrypted service related information to a server.

13. The method of claim 12 wherein communications between the first user and said first gate employ a first encryption key and communications between the second gate and the server employ a second encryption key.

14. The method of claim 12 wherein said step of authenticating said first user comprises verifying that stored account records reflect that the first user is a network subscriber service.

15. The method of claim 14 wherein said step of determining whether said first user is authorized to access the service includes the step of examining stored account records reflecting said first user's service access privileges.

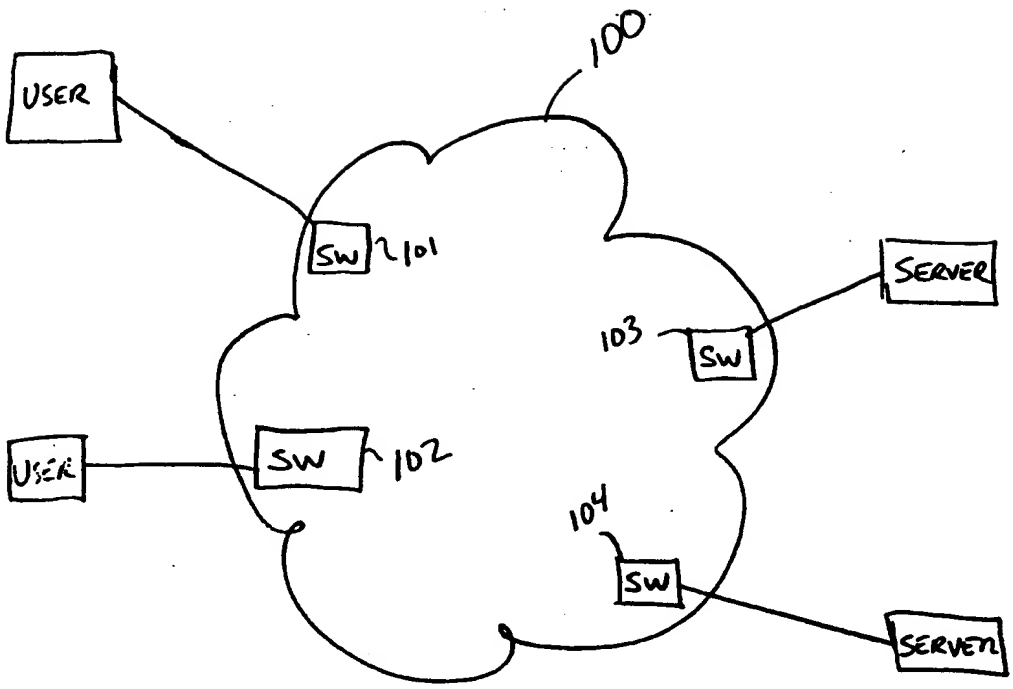


FIG. 1
(Prior Art)

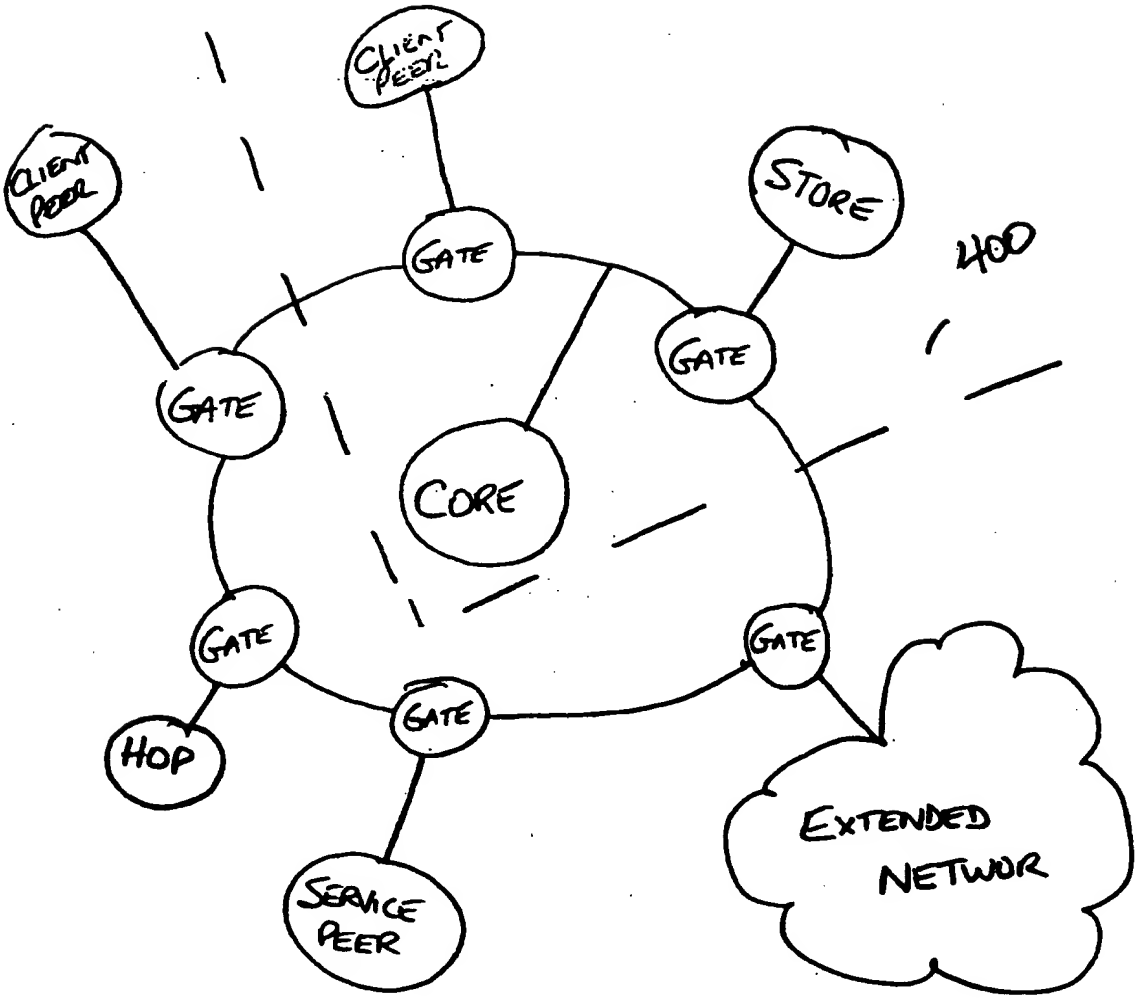

CLIENT PEER
CLIENT PEER
STORE
400
GATE
GATE
GATE
CORE
GATE
GATE
GATE
HOP
SERVICE PEER
EXTENDED NETWOR

FIG. 2

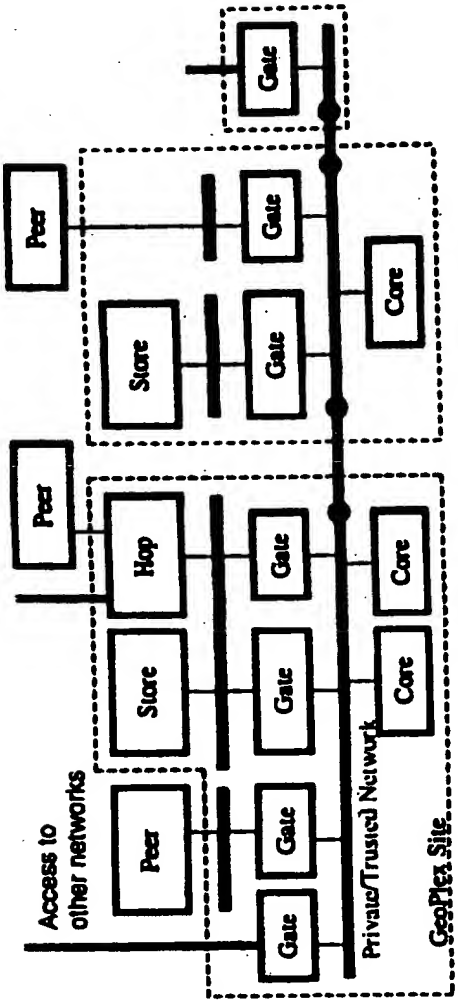

Access to other networks
Peer
Peer
Peer
Hop
Store
Store
Gate
Gate
Gate
Gate
Gate
Gate
Gate
Core
Core
Core
Private/Trusted Network
GeoPlex Site

FIG. 3

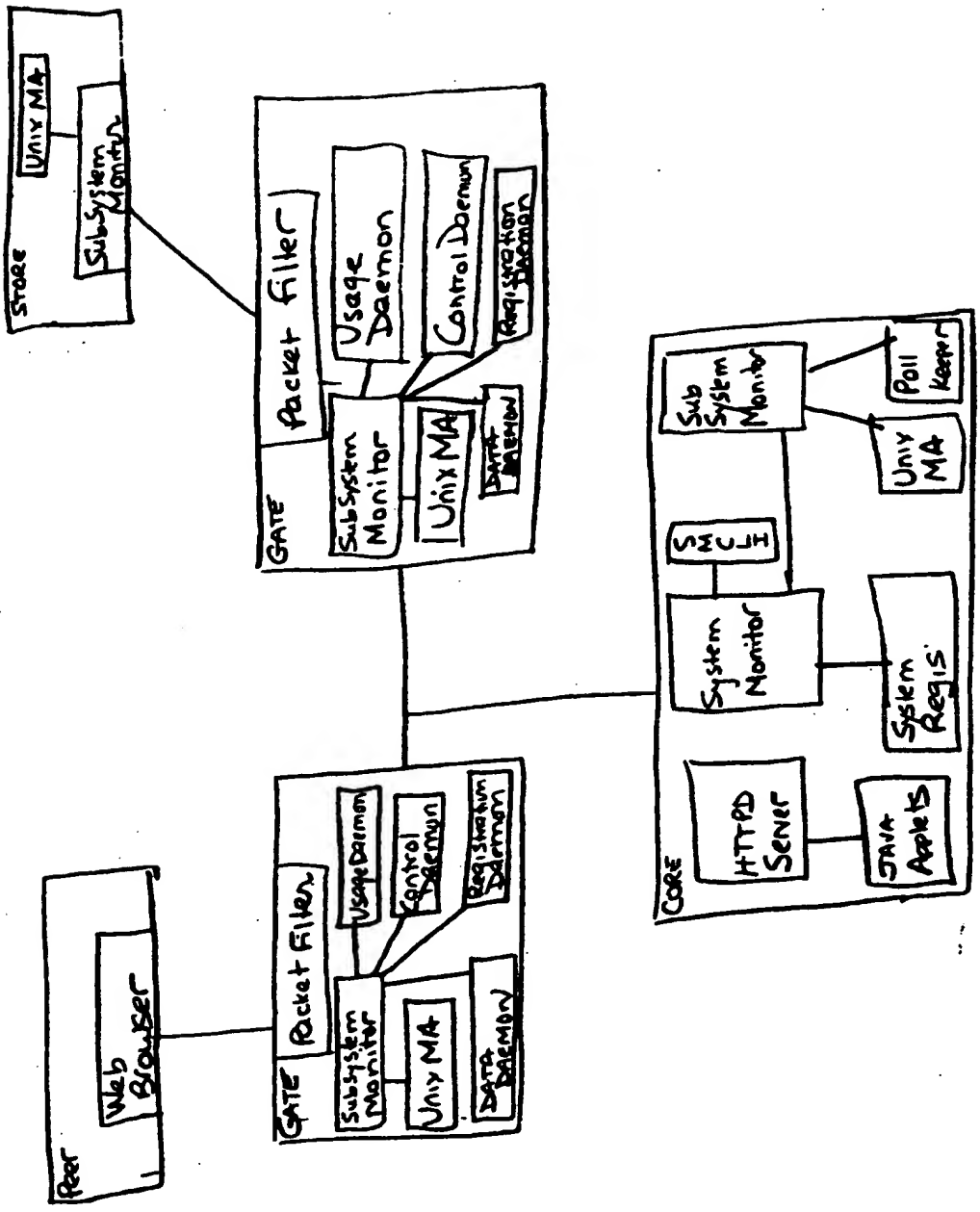

Store
Univ MA
SubSystem Monitor
GATE
Packet Filter
Usage Daemon
Control Daemon
Registration Daemon
SubSystem Monitor
Univ MA
Data Daemon
CORE
Sub System Monitor
Poll Keeper
Univ MA
SNMP
System Monitor
System Regis
HTTPD Sever
JAVA Applets
Peer
Web Browser
GATE
Packet Filter
Usage Daemon
Control Daemon
Registration Daemon
SubSystem Monitor
Univ MA
Data Daemon

FIG. 4

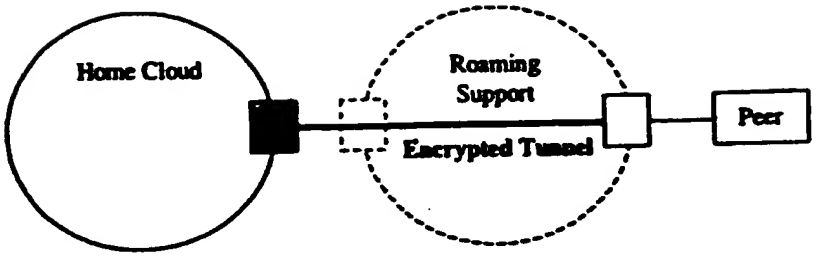

Home Cloud
Roaming
Support
Encrypted Tunnel
Peer

FIG. 5

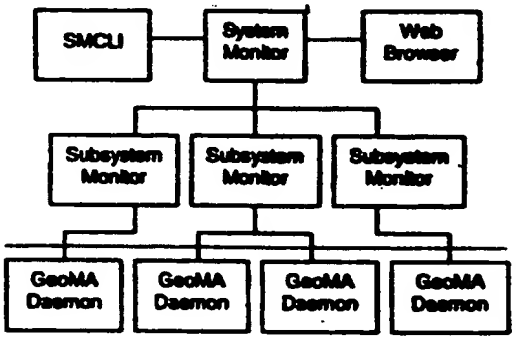

SMCLI
System Monitor
Web Browser
Subsystem Monitor
Subsystem Monitor
Subsystem Monitor
GeoMA Daemon
GeoMA Daemon
GeoMA Daemon
GeoMA Daemon

FIG. 6

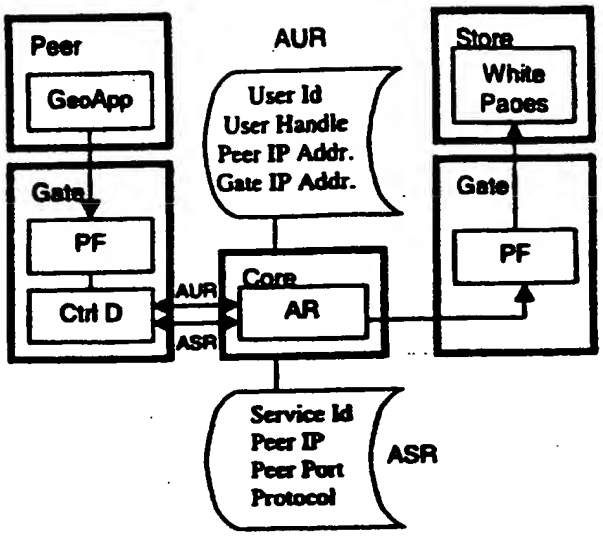

Peer
GeoApp
Gate
PF
Ctrl D
AUR
ASR
AUR
User Id
User Handle
Peer IP Addr.
Gate IP Addr.
Core
AR
Service Id
Peer IP
Peer Port
Protocol
ASR
Store
White Pages
Gate
PF

FIG. 7

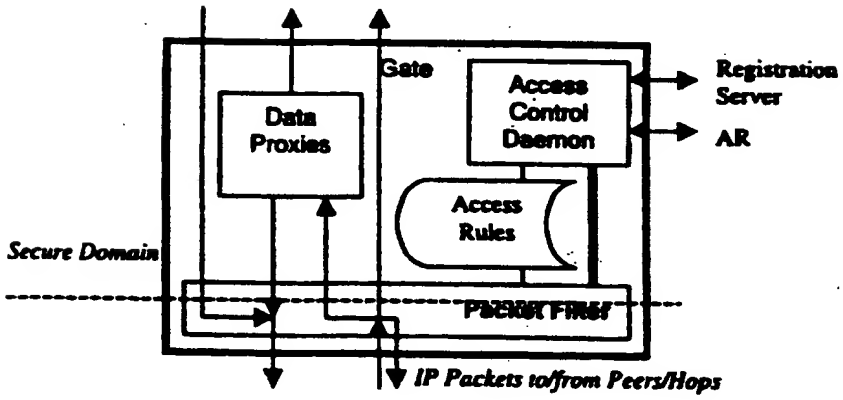

Gate
Data Proxies
Access Control Daemon
Registration Server
AR
Access Rules
Secure Domain
IP Packets to/from Peers/Hops

FIG. 8

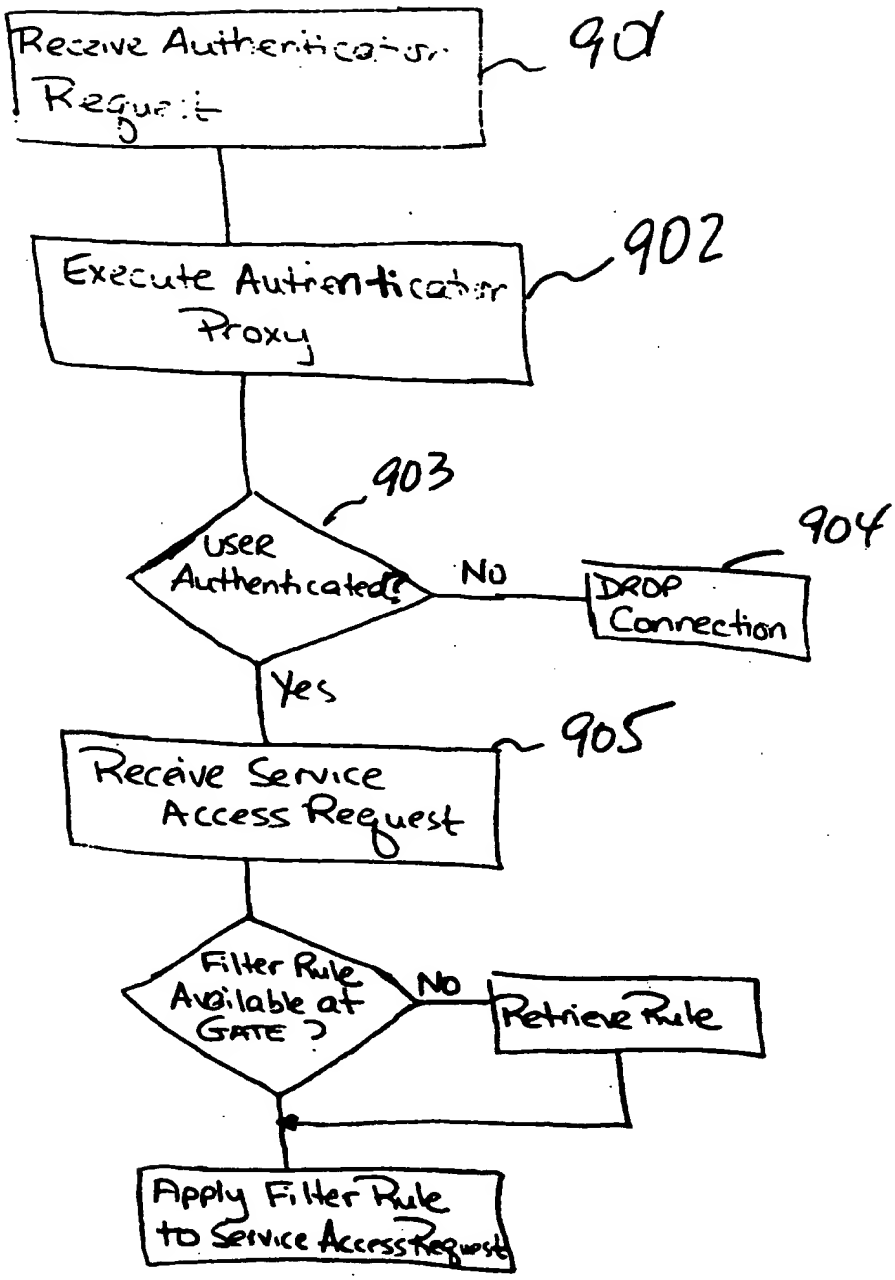

Receive Authenticator Request
901
Execute Authenticator Proxy
902
903
User Authenticated?
No
904
DROP Connection
Yes
Receive Service Access Request
905
Filter Rule Available at GATE ?
No
Retrieve Rule
Apply Filter Rule to Service Access Request

FIG. 9

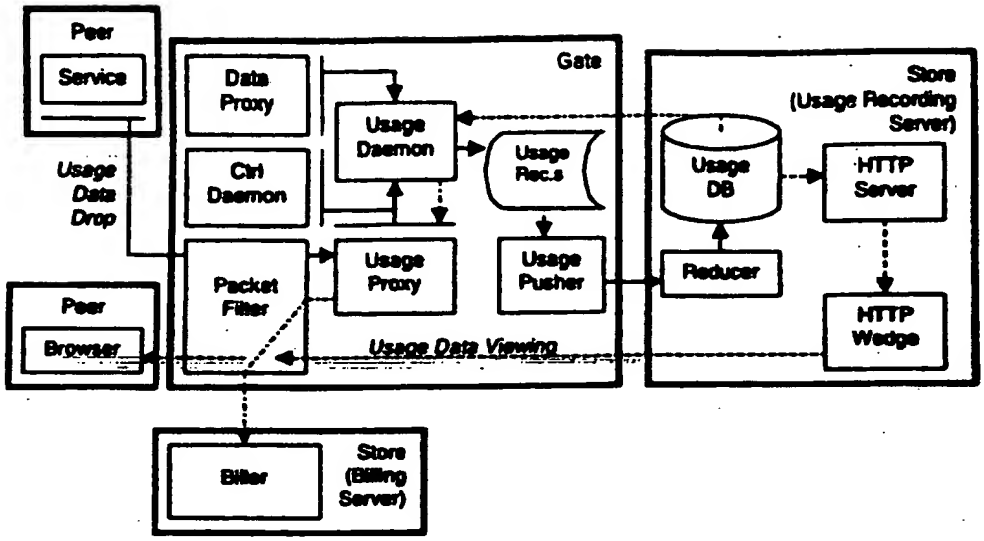

Peer
Service
Usage Data Drop
Peer
Browser
Gate
Data Proxy
Ctrl Daemon
Usage Daemon
Usage Rec.s
Packet Filter
Usage Proxy
Usage Pusher
Usage Data Viewing
Store (Usage Recording Server)
Usage DB
HTTP Server
Reducer
HTTP Wedge
Biller
Store (Billing Server)

FIG. 10

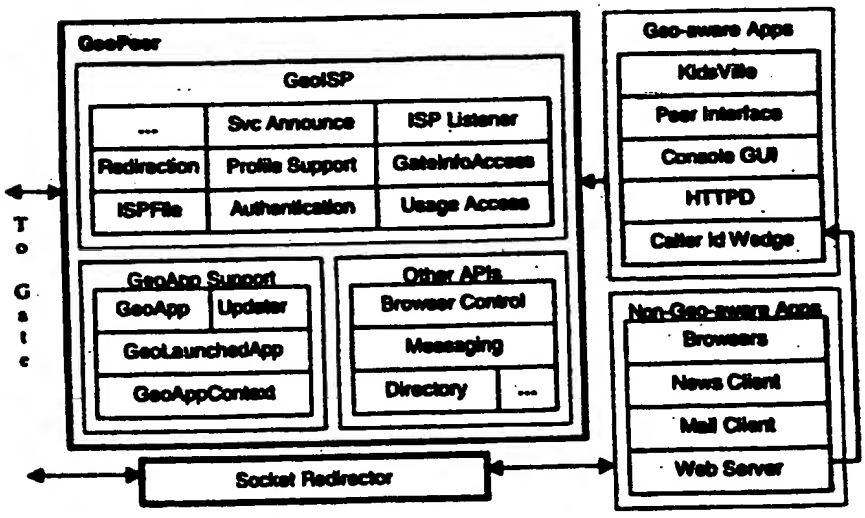

GeoPeer
GeoISP
...
Svc Announce
ISP Listener
Redirection
Profile Support
GateInfoAccess
ISPFile
Authentication
Usage Access
GeoApp Support
GeoApp
Updater
GeoLaunchedApp
GeoAppContext
Other APIs
Browser Control
Messaging
Directory
...
To Gate
Socket Redirector
Geo-aware Apps
KidsVille
Peer Interface
Console GUI
HTTPD
Caller Id Wedge
Non-Geo-aware Apps
Browsers
News Client
Mail Client
Web Server

FIG. 11

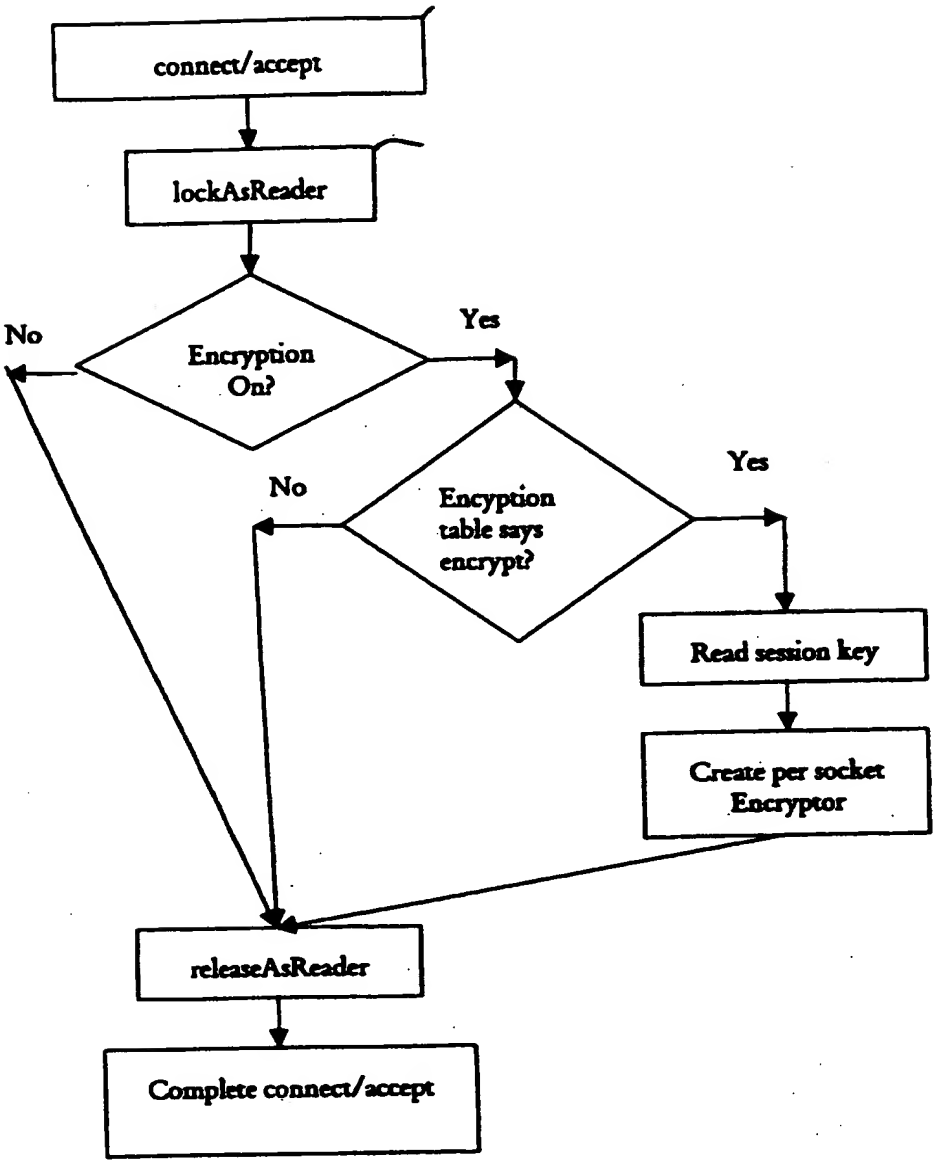

connect/accept
lockAsReader
Encryption On?
No
Yes
Encyption table says encrypt?
No
Yes
Read session key
Create per socket Encryptor
releaseAsReader
Complete connect/accept

FIG. 13

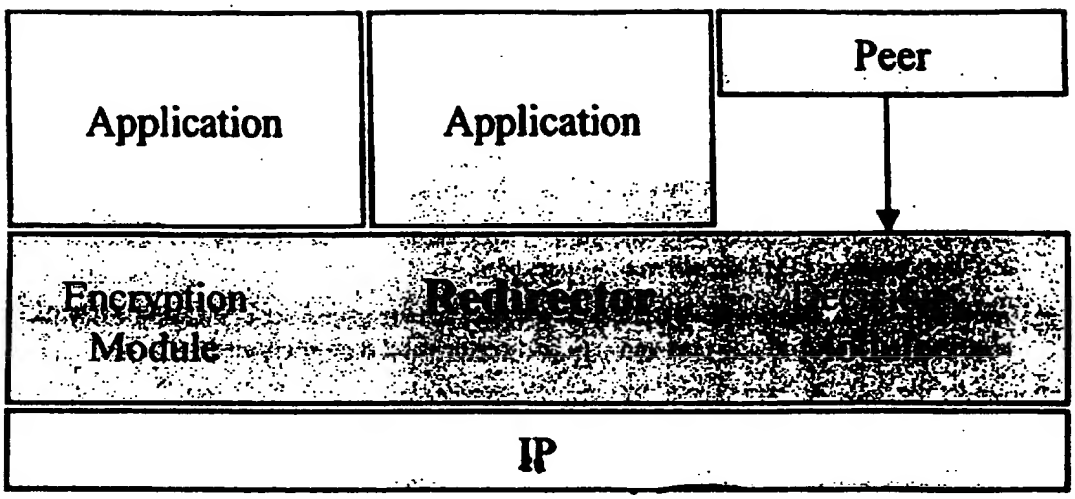

Application
Application
Peer
Encryption Module
IP

FIG. 12

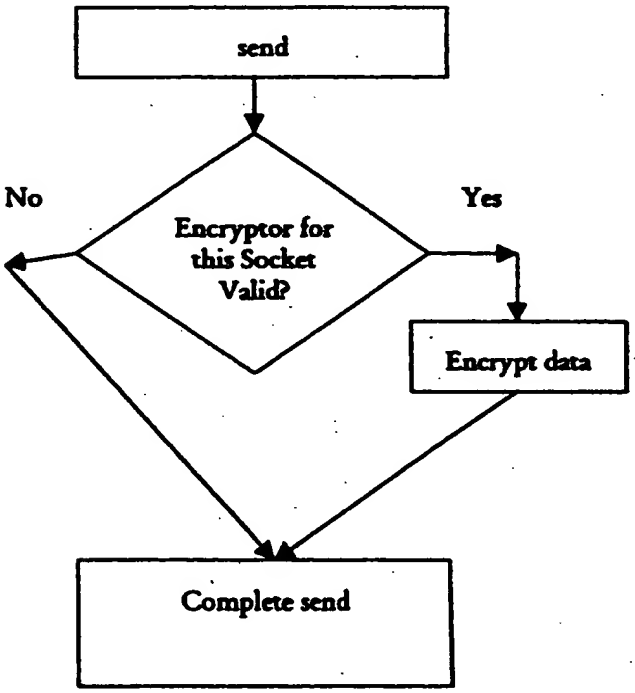

send
No
Encryptor for this Socket Valid?
Yes
Encrypt data
Complete send

FIG. 14

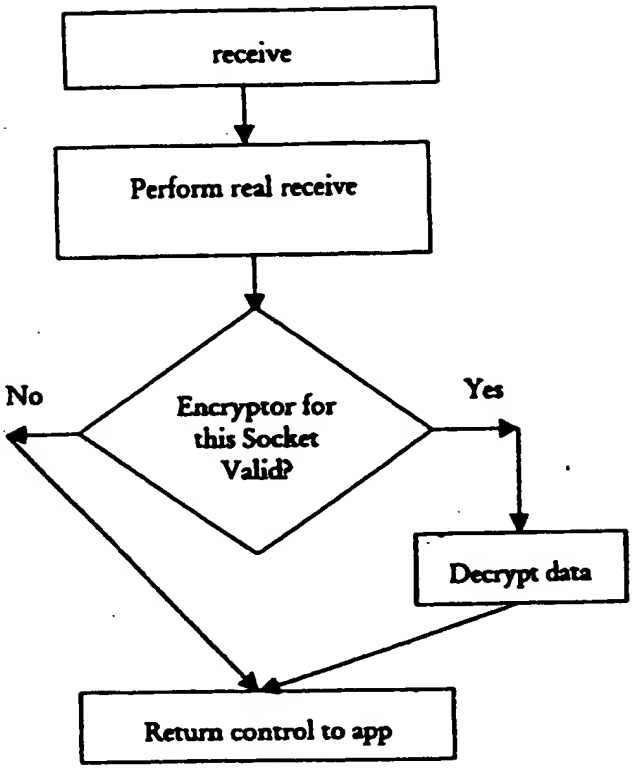

receive
Perform real receive
Encryptor for this Socket Valid?
No
Yes
Decrypt data
Return control to app

FIG. 15

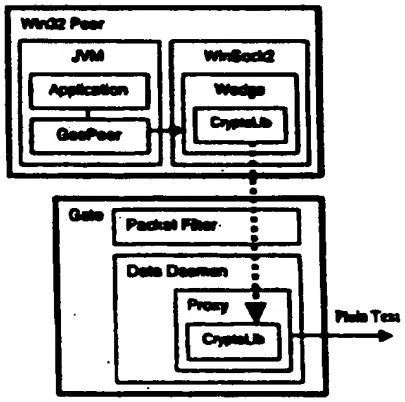

Win32 Peer
JVM
Application
GssPeer
WinSock2
Wedge
CryptoLib
Gate
Packet Filter
Data Daemon
Proxy
CryptoLib
Plain Text

FIG. 16

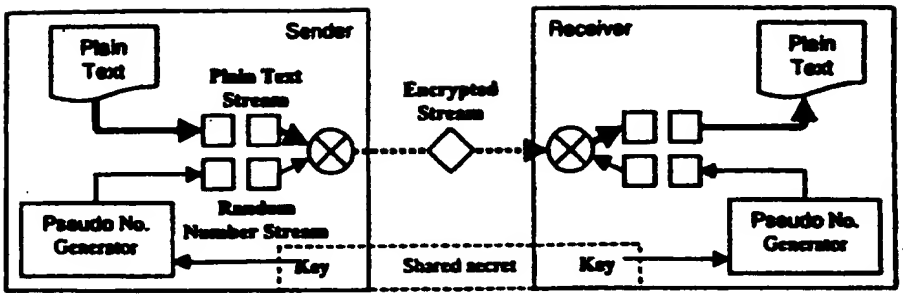

Sender
Plain Text
Plain Text Stream
Encrypted Stream
Receiver
Plain Text
Pseudo No. Generator
Random Number Stream
Key
Shared secret
Key
Pseudo No. Generator

FIG. 17

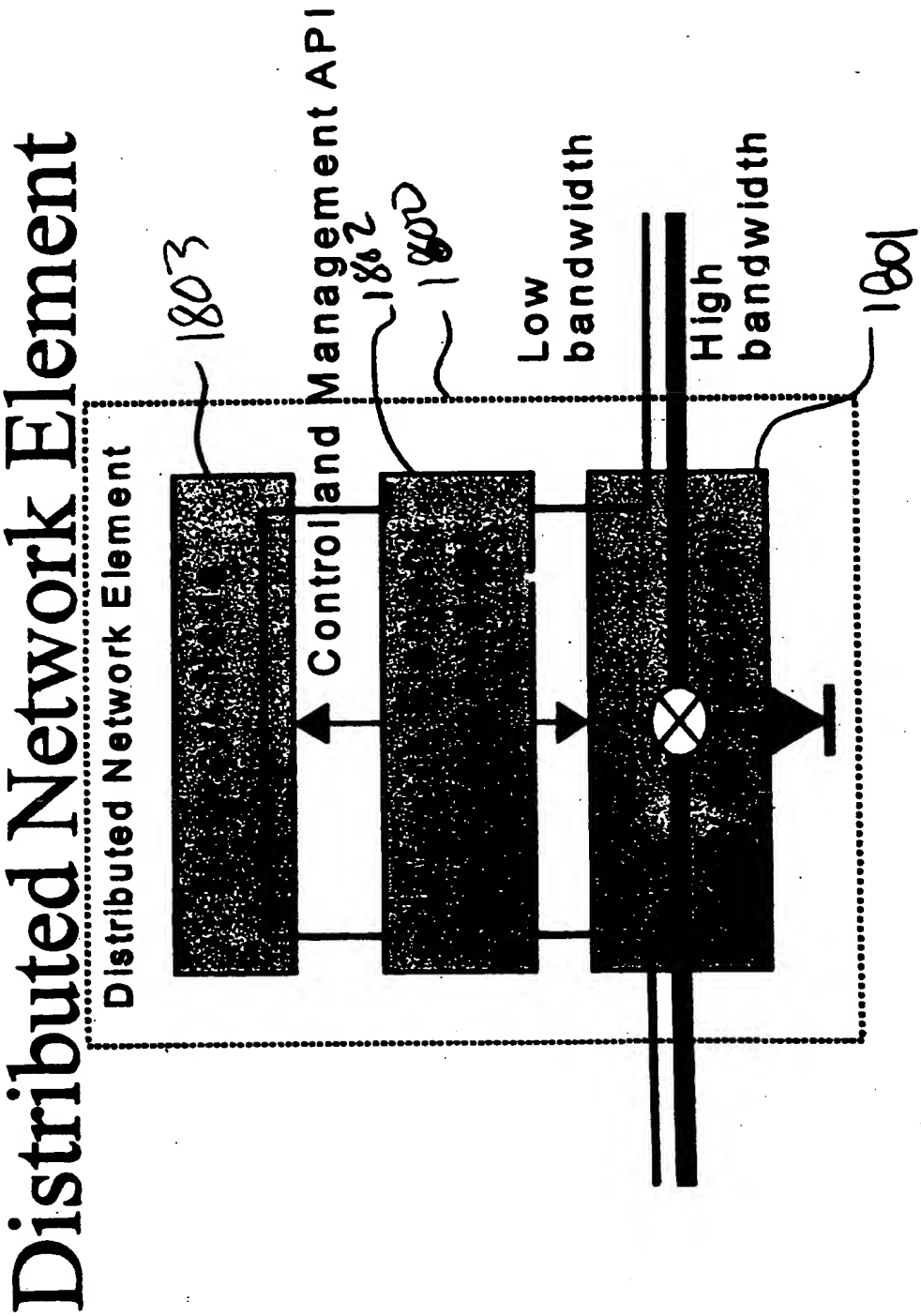

Distributed Network Element
Distributed Network Element
1803
Control and Management API
1802
1800
Low bandwidth
High bandwidth
1801

FIG. 18

New Architecture of a GeoPlex

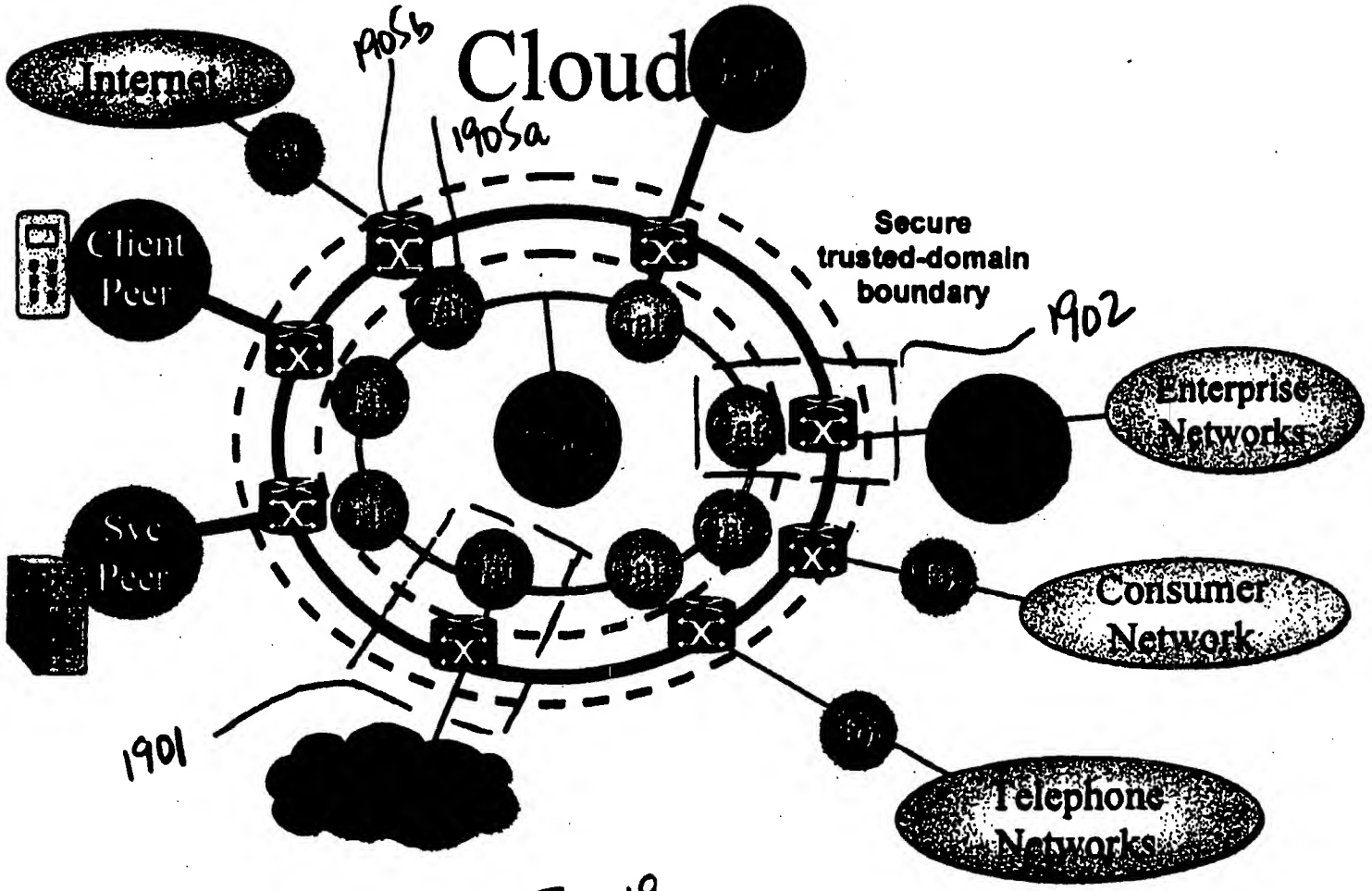


FIG. 19

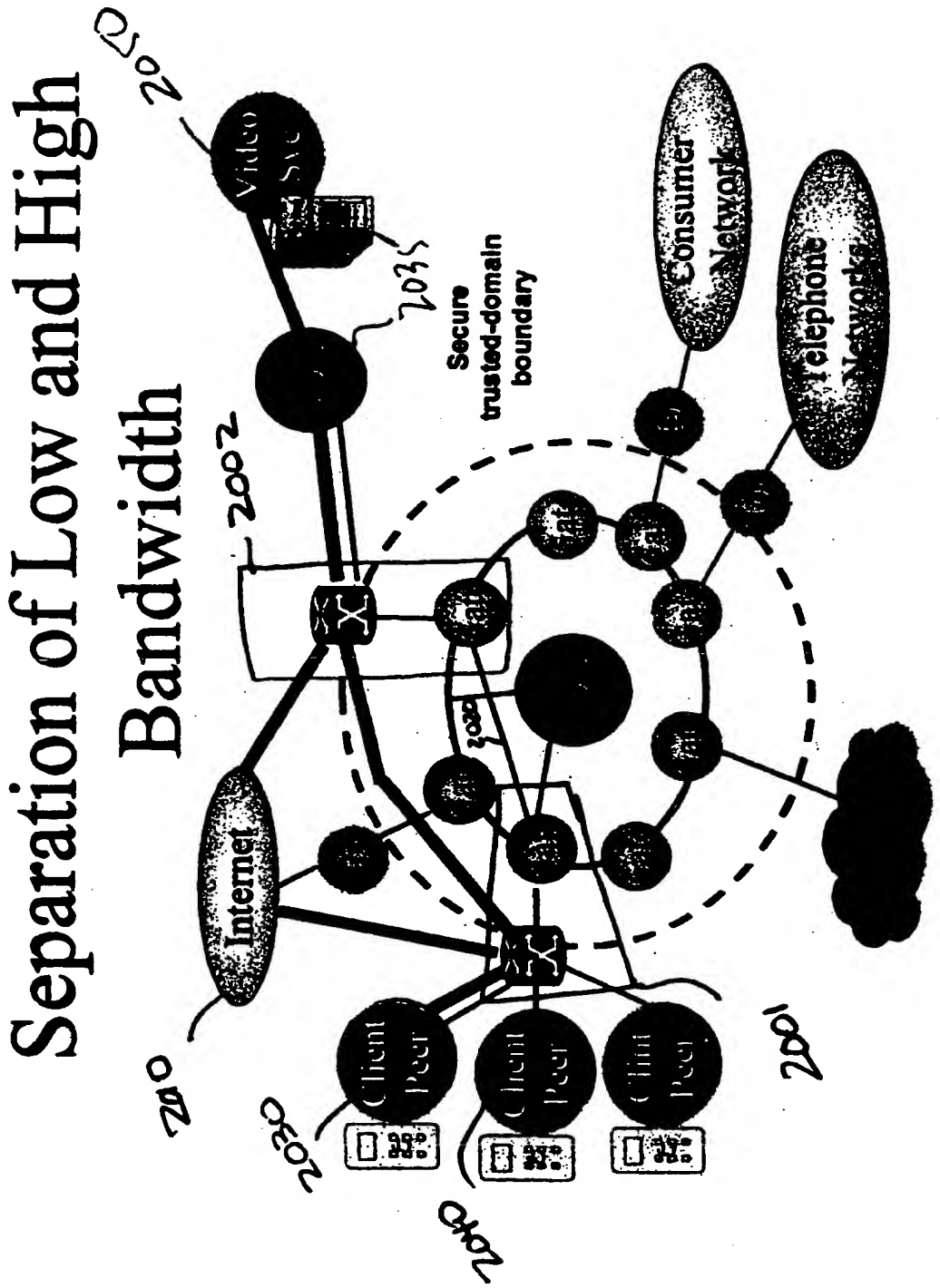

Separation of Low and High Bandwidth
Video SVC
Internet
Client Peer
Client Peer
Client Peer
Consumer Network
Telephone Networks
Secure trusted-domain boundary
2001
2002
2003
2004
2005
2006
2007

FIG 20

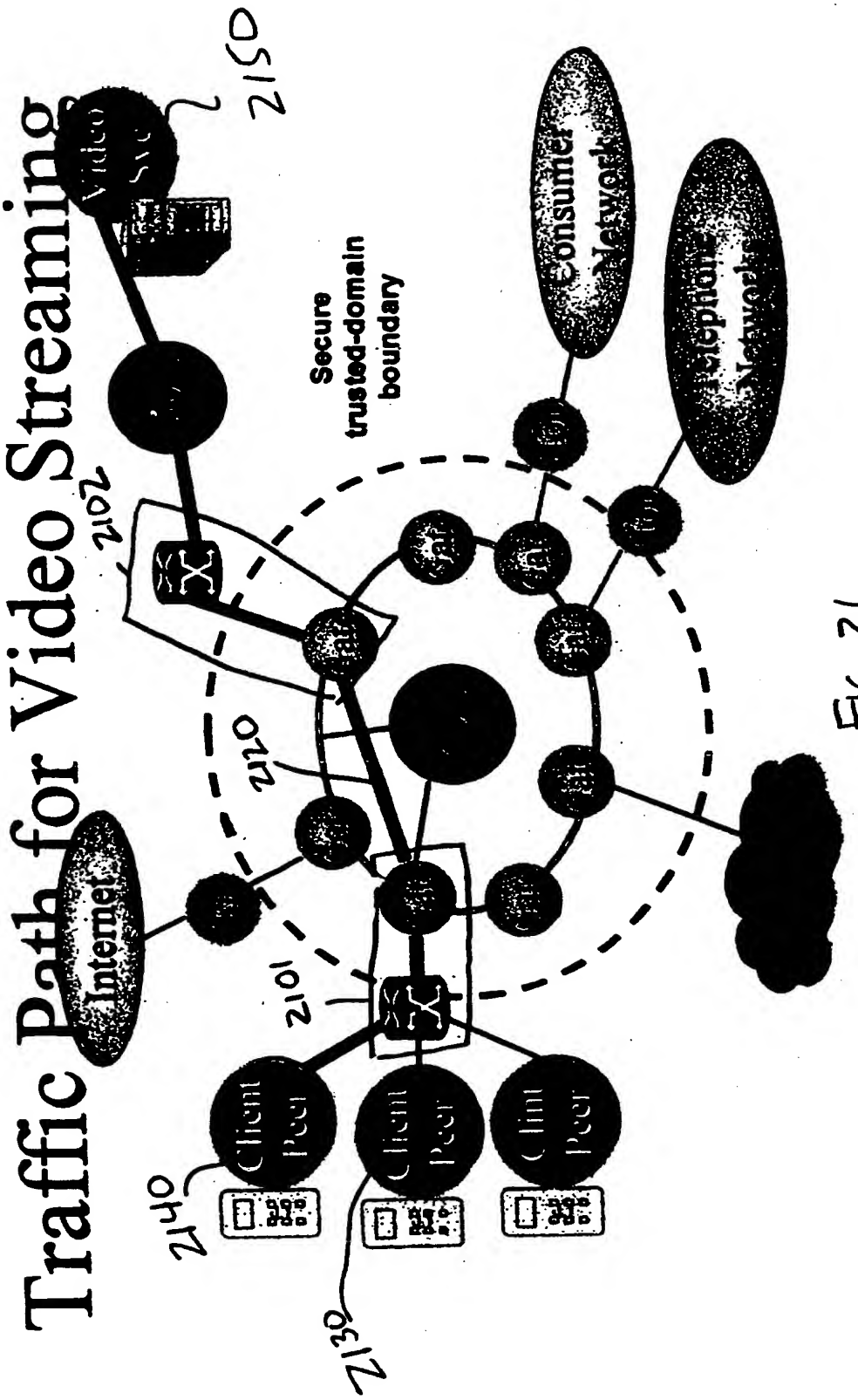

Traffic Path for Video Streaming
Video Svc
2150
2102
Secure trusted-domain boundary
Consumer Network
Telephone Network
Internet
2120
2101
Client Peer
Client Peer
Client Peer
2140
2130

Fig 21